Scenario: Creating an XML file using a loop......................................................................................................................... 114

**tAggregateRow** ............................................................................................................................................................................ 119
  tAggregateRow Standard properties........................................................................................................................................... 119
  Aggregating values and sorting data.......................................................................................................................................... 121

**tAggregateSortedRow** .............................................................................................................................................................. 125
  tAggregateSortedRow Standard properties............................................................................................................................... 125
  Scenario: Sorting and aggregating the input data....................................................................................................................... 127

**tAmazonAuroraCommit** .............................................................................................................................................................. 134
  tAmazonAuroraCommit Standard properties............................................................................................................................. 134
  Related scenario.......................................................................................................................................................................... 135

**tAmazonAuroraClose** ............................................................................................................................................................... 132
  tAmazonAuroraClose Standard properties................................................................................................................................ 132

**tAmazonAuroraConnection** ....................................................................................................................................................... 136
  tAmazonAuroraConnection Standard properties....................................................................................................................... 136
  Related scenario.......................................................................................................................................................................... 138

**tAmazonAuroraInput** ............................................................................................................................................................... 139
  tAmazonAuroraInput Standard properties................................................................................................................................. 139
  Scenario: Handling data with Amazon Aurora......................................................................................................................... 142

**tAmazonAuroraOutput** .............................................................................................................................................................. 149
  tAmazonAuroraOutput Standard properties............................................................................................................................... 149
  Related scenario.......................................................................................................................................................................... 155

**tAmazonAuroraRollback** ......................................................................................................................................................... 156
  tAmazonAuroraRollback Standard properties............................................................................................................................ 156
  Related Scenario........................................................................................................................................................................ 157

**tAmazonEMRListInstances** ....................................................................................................................................................... 158
  tAmazonEMRListInstances Standard properties....................................................................................................................... 158
  Related scenario.......................................................................................................................................................................... 159

**tAmazonEMRManage** .............................................................................................................................................................. 160
  tAmazonEMRManage Standard properties................................................................................................................................. 160
  Managing an Amazon EMR cluster............................................................................................................................................ 164

**tAmazonEMRResize** ................................................................................................................................................................. 169
  tAmazonEMRResize Standard properties................................................................................................................................. 169
  Related scenario.......................................................................................................................................................................... 171

**tAmazonMysqlClose** ............................................................................................................................................................... 172
  tAmazonMysqlClose Standard properties................................................................................................................................. 172
  Related scenarios....................................................................................................................................................................... 173

**tAmazonMysqlCommit** ............................................................................................................................................................. 174
  tAmazonMysqlCommit Standard properties............................................................................................................................... 174
  Related scenario.......................................................................................................................................................................... 175

**tAmazonMysqlConnection** ...................................................................................................................................................... 176
  tAmazonMysqlConnection Standard properties....................................................................................................................... 176
  Related scenario.......................................................................................................................................................................... 178

**tAmazonMysqlInput** ............................................................................................................................................................... 179
  tAmazonMysqlInput Standard properties................................................................................................................................. 179
Related scenarios........................................................................................................................................................................182

**tAmazonMysqlOutput**........................................................................................................................................................................183
tAmazonMysqlOutput Standard properties.................................................................................................................................183
Related scenarios...........................................................................................................................................................................188

**tAmazonMysqlRollback**....................................................................................................................................................................189
tAmazonMysqlRollback Standard properties................................................................................................................................189
Related scenario.................................................................................................................................................................................190

**tAmazonMysqlRow**.............................................................................................................................................................................191
tAmazonMysqlRow Standard properties...........................................................................................................................................191
Related scenario................................................................................................................................................................................201

**tAmazonOracleClose**..........................................................................................................................................................................195
tAmazonOracleClose Standard properties.......................................................................................................................................195
Related scenario..................................................................................................................................................................................196

**tAmazonOracleCommit**.....................................................................................................................................................................197
tAmazonOracleCommit Standard properties...................................................................................................................................197
Related scenario..................................................................................................................................................................................198

**tAmazonOracleConnection**..............................................................................................................................................................199
tAmazonOracleConnection Standard properties................................................................................................................................199
Related scenario..................................................................................................................................................................................201

**tAmazonOracleInput**....................................................................................................................................................................202
tAmazonOracleInput Standard properties........................................................................................................................................202
Related scenario....................................................................................................................................................................................205

**tAmazonOracleOutput**.....................................................................................................................................................................206
tAmazonOracleOutput Standard properties......................................................................................................................................206
Related scenario....................................................................................................................................................................................211

**tAmazonOracleRollback**.................................................................................................................................................................212
tAmazonOracleRollback Standard properties..................................................................................................................................212
Related scenario....................................................................................................................................................................................212

**tAmazonOracleRow**.............................................................................................................................................................................214
tAmazonOracleRow Standard properties...........................................................................................................................................214
Related scenario....................................................................................................................................................................................213

**tAmazonRedshiftManage**.................................................................................................................................................................218
tAmazonRedshiftManage Standard properties..................................................................................................................................218
Related scenario....................................................................................................................................................................................217

**tApacheLogInput**.............................................................................................................................................................................222
tApacheLogInput Standard properties.............................................................................................................................................222
Scenario: Reading an Apache access-log file..................................................................................................................................223

**tAS400Close**.....................................................................................................................................................................................225
tAS400Close Standard properties...................................................................................................................................................225
Related scenario....................................................................................................................................................................................226

**tAS400Commit**..................................................................................................................................................................................227
tAS400Commit Standard properties..................................................................................................................................................227
Related scenario....................................................................................................................................................................................228

**tAS400Connection**.............................................................................................................................................................................229
tAS400Connection Standard properties...........................................................................................................................................229
Scenario: Retrieving files from a Azure Storage container

- tAzureStorageGet
- tAzureStorageConnection
- tAzureStorageContainerCreate
- tAzureStorageContainerDelete
- tAzureStorageContainerList
- tAzureStorageContainerExist

Scenario 2: Setting up the assertive condition for a Job execution

- tAssert
- tAssertCatcher
- tAzureStorageConnection
- tAzureStorageContainerCreate
- tAzureStorageContainerDelete
- tAzureStorageContainerList
- tAzureStorageDelete
- tAzureStorageGet

Scenario: Handling data with AS/400

- tAS400Input
- tAS400Output
- tAS400Rollback
- tAS400Row

Related scenarios:

- tAS400Input Standard properties
- tAS400Output Standard properties
- tAS400Rollback Standard properties
- tAS400Row Standard properties
- tAzureStorageConnection Standard properties
- tAzureStorageContainerCreate Standard properties
- tAzureStorageContainerDelete Standard properties
- tAzureStorageContainerList Standard properties
- tAzureStorageDelete Standard properties
- tAzureStorageGet Standard properties

Scenario: Retrieving files from a Azure Storage container

- tAzureStorageGet Standard properties
- Scenario: Retrieving files from a Azure Storage container
tAzureStorageInputTable............................................................................................................................. 292
	tAzureStorageInputTable Standard properties.................................................................................................. 292
Handling data with Microsoft Azure Table storage........................................................................................ 295

tAzureStorageList............................................................................................................................................ 302
	tAzureStorageList Standard properties................................................................................................................ 302
Related scenario.................................................................................................................................................. 305

tAzureStorageOutputTable.................................................................................................................................. 306
	tAzureStorageOutputTable Standard properties.................................................................................................. 306
Related scenario.................................................................................................................................................. 309

tAzureStoragePut.................................................................................................................................................. 310
	tAzureStoragePut Standard properties.................................................................................................................. 310
Related scenario................................................................................................................................................ 312

tAzureStorageQueueCreate.................................................................................................................................. 313
	tAzureStorageQueueCreate Standard properties.................................................................................................. 313
Related scenario................................................................................................................................................ 314

tAzureStorageQueueDelete.................................................................................................................................. 315
	tAzureStorageQueueDelete Standard properties.................................................................................................. 315
Related scenario................................................................................................................................................ 316

tAzureStorageQueueInput.................................................................................................................................... 317
	tAzureStorageQueueInput Standard properties.................................................................................................... 317
Related scenario................................................................................................................................................ 320

tAzureStorageQueueInputLoop............................................................................................................................... 321
	tAzureStorageQueueInputLoop Standard properties.................................................................................................. 321
Related scenario................................................................................................................................................ 323

tAzureStorageQueueList...................................................................................................................................... 324
	tAzureStorageQueueList Standard properties........................................................................................................ 324
Related scenario................................................................................................................................................ 326

tAzureStorageQueueOutput.................................................................................................................................... 327
	tAzureStorageQueueOutput Standard properties.................................................................................................... 327
Related scenario................................................................................................................................................ 329

tAzureStorageQueuePurge...................................................................................................................................... 330
	tAzureStorageQueuePurge Standard properties...................................................................................................... 330
Related scenario................................................................................................................................................ 331

TBarChart.......................................................................................................................................................... 332
	TBarChart Standard properties............................................................................................................................... 332
Scenario: Creating a bar chart from the input data............................................................................................... 334

tBigQueryBulkExec............................................................................................................................................. 341
	tBigQueryBulkExec Standard properties.................................................................................................................. 341
Related Scenario.................................................................................................................................................. 344

tBigQueryInput..................................................................................................................................................... 345
	tBigQueryInput Standard properties..................................................................................................................... 345
Scenario: Performing a query in BigQuery........................................................................................................... 347

tBigQueryOutput................................................................................................................................................... 351
	tBigQueryOutput Standard properties................................................................................................................... 351
Scenario: Writing data in BigQuery.......................................................................................................................... 354
tBigQueryOutputBulk .................................................................................................................... 360
  tBigQueryOutputBulk Standard properties.......................................................................................... 360
  Related Scenario ................................................................................................................................. 360
  Scenario 1: Executing a Bonita process via a Talend Job ........................................................................... 369
  Scenario 2: Outputting the process instance UUID over the Row > Main link .................................................... 375
  tBoxConnection ........................................................................................................................................ 377
  tBoxConnection Standard properties ........................................................................................................ 377
  Related scenario ...................................................................................................................................... 377
  Scenario 2: Outputting the process instance UUID over the Row > Main link .................................................... 375
  tBonitaDeploy ....................................................................................................................................... 363
  tBonitaDeploy Standard properties .......................................................................................................... 363
  Related Scenario ................................................................................................................................. 363
  Scenario 1: Executing a Bonita process via a Talend Job ........................................................................... 369
  Scenario 2: Outputting the process instance UUID over the Row > Main link .................................................... 375
  tBonitaInstantiateProcess ....................................................................................................................... 366
  tBonitaInstantiateProcess Standard properties .......................................................................................... 366
  Scenario 1: Executing a Bonita process via a Talend Job ........................................................................... 369
  Scenario 2: Outputting the process instance UUID over the Row > Main link .................................................... 375
  tBufferInput .......................................................................................................................................... 394
  tBufferInput Standard properties .......................................................................................................... 394
  Related Scenario ....................................................................................................................................... 394
  Scenario: Retrieving bufferized data ........................................................................................................... 395
  Related Scenario ....................................................................................................................................... 395
  tBufferOutput ......................................................................................................................................... 398
  tBufferOutput Standard properties .......................................................................................................... 398
  Scenario 1: Buffering data ....................................................................................................................... 399
  Buffering data to be used as a source system ............................................................................................ 401
  Scenario 2: Buffering output data on the weapp server .............................................................................. 402
  Scenario 3: Calling a Job with context variables from a browser .................................................................. 405
  Scenario 4: Calling a Job exported as Webservice in another Job .................................................................... 407
  tCassandraBulkExec ............................................................................................................................... 410
  tCassandraBulkExec Standard properties ............................................................................................. 410
  Related scenarios ..................................................................................................................................... 410
  Scenario 1: Executing a Bonita process via a Talend Job ........................................................................... 369
  Scenario 2: Outputting the process instance UUID over the Row > Main link .................................................... 375
  tCassandraClose ....................................................................................................................................... 413
  tCassandraClose Standard properties ...................................................................................................... 413
  Related Scenario ....................................................................................................................................... 413
  Scenario 1: Executing a Bonita process via a Talend Job ........................................................................... 369
  Scenario 2: Outputting the process instance UUID over the Row > Main link .................................................... 375
  tCassandraConnection ............................................................................................................................ 414
  tCassandraConnection Standard properties ............................................................................................. 414
  Related Scenario ..................................................................................................................................... 414
  Scenario 1: Executing a Bonita process via a Talend Job ........................................................................... 369
  Scenario 2: Outputting the process instance UUID over the Row > Main link .................................................... 375
  Scenario 3: Calling a Job with context variables from a browser .................................................................. 405
  Scenario 4: Calling a Job exported as Webservice in another Job .................................................................... 407
  Scenario 5: Executing a Bonita process via a Talend Job ........................................................................... 369
  Scenario 6: Outputting the process instance UUID over the Row > Main link .................................................... 375
Scenario: Filtering and aggregating table columns directly on the DBMS ......................................................... 467
Related scenarios............................................................................................................................................................ 467

Scenario: Measuring the processing time of a subjob and part of a subjob ...................................................... 455
Related scenario........................................................................................................................................................................ 455

Scenario: Transforming the character encoding of a file .......................................................................................... 451
Related scenarios............................................................................................................................................................ 451

Scenario: Handling data with Cassandra ......................................................................................................................... 421
Related scenario........................................................................................................................................................................ 421

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480

Related Scenario....................................................................................................................................................................... 480
tCouchDBOutput..............................................................................................................................531
  tCouchDBOutput Standard properties............................................................................................531
  Related scenario............................................................................................................................532

Scenario: Creating new table in a Mysql Database .....................................................................................................545
  tCreateTable Standard properties........................................................................................................541
  Related scenario............................................................................................................................542

Related Scenario....................................................................................................................................................................... 530
  tCouchDBInput Standard properties........................................................................................................526
  Related scenarios........................................................................................................................................523

Scenario: Replicating data from the source database to the target database ...................................................534
  tCouchDBClose Standard properties............................................................................................................524
  Related Scenario........................................................................................................................................524

Related scenario........................................................................................................................................................................484
  tCombinedSQLInput Standard properties..........................................................................................481
  Related scenario............................................................................................................................482
  tCombinedSQLOutput Standard properties..........................................................................................483
  Related scenario............................................................................................................................484

Scenario: Converting java types ..........................................................................................................................................496
  tConvertType Standard properties........................................................................................................495
  Scenario: Converting java types.............................................................................................................496

Related scenario........................................................................................................................................................................482
  tCombinedSQLInput Standard properties..........................................................................................481
  Related scenario............................................................................................................................482
  tCombinedSQLOutput Standard properties..........................................................................................483
  Related scenario............................................................................................................................484

Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters..........................................................................................................................................489
  tConvertType Standard properties........................................................................................................495
  Scenario: Converting java types.............................................................................................................496

Related scenario........................................................................................................................................................................484
  tCombinedSQLInput Standard properties..........................................................................................481
  Related scenario............................................................................................................................482
  tCombinedSQLOutput Standard properties..........................................................................................483
  Related scenario............................................................................................................................484

tCosmosDBBulkLoad...........................................................................................................................................501
  tCosmosDBBulkLoad Standard properties..........................................................................................501

Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters..........................................................................................................................................489
  tConvertType Standard properties........................................................................................................495
  Scenario: Converting java types.............................................................................................................496

Related scenario........................................................................................................................................................................484
  tCombinedSQLInput Standard properties..........................................................................................481
  Related scenario............................................................................................................................482
  tCombinedSQLOutput Standard properties..........................................................................................483
  Related scenario............................................................................................................................484

ContextDump..........................................................................................................................................................485
  ContextDump Standard properties........................................................................................................485
  Related scenarios......................................................................................................................................486

Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters..........................................................................................................................................489
  tConvertType Standard properties........................................................................................................495
  Scenario: Converting java types.............................................................................................................496

Related scenario........................................................................................................................................................................484
  tCombinedSQLInput Standard properties..........................................................................................481
  Related scenario............................................................................................................................482
  tCombinedSQLOutput Standard properties..........................................................................................483
  Related scenario............................................................................................................................484

tContextLoad...........................................................................................................................................................................487
  tContextLoad Standard properties........................................................................................................487
  Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters..........................................................................................................................................489

Related scenario........................................................................................................................................................................484
  tCombinedSQLInput Standard properties..........................................................................................481
  Related scenario............................................................................................................................482
  tCombinedSQLOutput Standard properties..........................................................................................483
  Related scenario............................................................................................................................484

tConvertType...........................................................................................................................................................................495
  tConvertType Standard properties........................................................................................................495
  Scenario: Converting java types.............................................................................................................496

Related scenario........................................................................................................................................................................484
  tCombinedSQLInput Standard properties..........................................................................................481
  Related scenario............................................................................................................................482
  tCombinedSQLOutput Standard properties..........................................................................................483
  Related scenario............................................................................................................................484

Parameter Load.........................................................................................................................................................489
Scenario: Utilizing .NET in Talend

Related scenarios

Scenario: Regrouping sorted rows

Scenario: Resetting a DB auto-increment

Related scenarios

Scenario: Utilizing .NET in Talend
Scenario: Joining table columns and writing them into Hive

**tELTHiveMap**

- Standard properties
- Related scenarios

Scenario: Joining table columns and writing them into Hive

**tELTHiveInput**

- Standard properties
- Related scenarios

Scenario: Joining table columns and writing them into Hive

**tELTGreenplumOutput**

- Standard properties
- Related scenarios

Scenario: Joining table columns and writing them into Hive

**tELTGreenplumMap**

- Standard properties
- Related scenarios

Scenario: Joining table columns and writing them into Hive

**tELTGreenplumInput**

- Standard properties
- Related scenarios

Scenario: Joining table columns and writing them into Hive

**tEDIFACTtoXML**

- Standard properties
- Related scenarios

Scenario: From EDIFACT to XML

**tDTDValidator**

- Standard properties
- Related scenarios

Scenario: Validating XML files

**tDynamoDBInput**

- Standard properties
- Related scenarios

Scenario: Validating XML files

**tDynamoDBOutput**

- Standard properties
- Related scenarios

Scenario: Validating XML files

**tEDIFACTtoXML**

- Standard properties
- Related scenarios

Scenario: Validating XML files

**tELTGreenplumInput**

- Standard properties
- Related scenarios

Scenario: Validating XML files

**tELTGreenplumMap**

- Standard properties
- Related scenarios

Scenario: Mapping data using a simple implicit join

**tELTGreenplumOutput**

- Standard properties
- Related scenarios

Scenario: Joining table columns and writing them into Hive

**tELTHiveInput**

- Standard properties
- Related scenarios

Scenario: Joining table columns and writing them into Hive

**tELTHiveMap**

- Standard properties
- Related scenarios

Scenario: Joining table columns and writing them into Hive

**tELTHiveOutput**

- Standard properties
- Related scenarios
Scenario 1: Using tESBConsumer to retrieve the valid email.......................................................... 825
Scenario 2: Using tESBConsumer with custom SOAP Headers....................................................... 833
tESBProviderFault............................................................................................................................ 844
tESBProviderFault Standard properties.......................................................................................... 844
Scenario: Requesting airport names based on country codes....................................................... 845
tESBProviderRequest....................................................................................................................... 857
tESBProviderRequest Standard properties...................................................................................... 857
Scenario: Service sending a message without expecting a response................................................ 860
tESBProviderResponse.................................................................................................................... 870
tESBProviderResponse Standard properties.................................................................................. 870
Scenario: Returning Hello world response..................................................................................... 871
tEXABulkExec................................................................................................................................. 882
tEXABulkExec Standard properties.................................................................................................. 882
Settings for different sources of import data................................................................................... 887
Scenario: Importing data into an EXASolution database table from a local CSV file..................... 890
tEXAClose........................................................................................................................................... 897
tEXAClose Standard properties......................................................................................................... 897
Related scenario.............................................................................................................................. 898
tEXACommit.................................................................................................................................... 899
tEXACommit Standard properties..................................................................................................... 899
Related scenario.............................................................................................................................. 900
tEXAConnection.............................................................................................................................. 901
tEXAConnection Standard properties............................................................................................. 901
Related scenario.............................................................................................................................. 903
tEXAInput......................................................................................................................................... 904
tEXAInput Standard properties......................................................................................................... 904
Related scenario.............................................................................................................................. 907
tEXAOutput...................................................................................................................................... 908
tEXAOutput Standard properties....................................................................................................... 908
Related scenario.............................................................................................................................. 913
tEXARollback................................................................................................................................... 914
tEXARollback Standard properties.................................................................................................... 914
Related Scenario.............................................................................................................................. 915
tEXARow.......................................................................................................................................... 916
tEXARow Standard properties........................................................................................................... 916
Related Scenario.............................................................................................................................. 919
tEXistConnection............................................................................................................................. 920
tEXistConnection Standard properties........................................................................................... 920
Related scenarios............................................................................................................................. 921
tEXistDelete.................................................................................................................................... 922
tEXistDelete Standard properties...................................................................................................... 922
Related scenarios............................................................................................................................. 923
tEXistGet.......................................................................................................................................... 924
tEXistGet Standard properties........................................................................................................... 924
Scenario: Retrieve resources from a remote eXist DB server............................................................ 925
Scenario: Deleting files

- tFileDelete Standard properties
- Related scenarios

Scenario: Restoring files from bin

- tFileCopy Standard properties
- Related scenarios

Scenario: Comparing unzipped files

- tFileCompare Standard properties
- Related scenarios

Scenario 2: Extracting correct and erroneous data from an XML field in a delimited file

- tExtractDelimitedFields Standard properties
- Scenario: Extracting a delimited string column of a database table

Scenario 1: Extracting XML data from a field in a database table

- tExtractXMLField Standard properties
- Scenario 1: Extracting XML data from a field in a database table
- Scenario 2: Extracting correct and erroneous data from an XML field in a delimited file

Scenario: Extracting name, domain and TLD from e-mail addresses

- tExtractPositionalFields Standard properties
- Related scenario

- tExtractRegexFields Standard properties
- Scenario: Extracting name, domain and TLD from e-mail addresses

Scenario: Extracting XML field

- tExtractXMLField Standard properties
- Scenario 1: Extracting XML data from a field in a database table
- Scenario 2: Extracting correct and erroneous data from an XML field in a delimited file

Scenario: Archiving files

- tFileArchive Standard properties
- Scenario: Zip files using a tFileArchive

Scenario: Comparing unzipped files

- tFileCompare Standard properties
- Scenario: Comparing unzipped files

Scenario: Copying files

- tFileCopy Standard properties
- Scenario: Restoring files from bin

Scenario: Deleting files

- tFileDelete Standard properties
- Scenario: Deleting files

Scenario: Checking file existence

- tFileExist Standard properties
tFileInputProperties.................................................................................................................................................. 1075
   tFileInputProperties Standard properties.............................................................................................................. 1075
   Scenario: Reading and matching the keys and the values of different .properties files and outputting
   the results in a glossary............................................................................................................................................... 1076

Scenario: Writing data from a database table into an LDIF file.............................................................................. 1032
   tFileOutputLDIF Standard properties............................................................................................................................. 1029
   Related Scenario........................................................................................................................................................... 1030
   Scenario: Writing a JSON structured file...................................................................................................................1025
   tFileOutputJSON Standard properties.............................................................................................................................1024
   Related Scenario................................................................................................................................................................... 1023
   Scenario 2: Utilizing Output Stream to save filtered data to a local file......................................................... 1117
   tFileOutputDelimited Standard properties................................................................................................................... 1110
   Related Scenario..................................................................................................................................................................... 1109
   tFileOutputARFF Standard properties.............................................................................................................................. 1107
   Related scenario..................................................................................................................................................................... 1106
   Scenario 1: Iterating on a file directory......................................................................................................................1099
   Scenario 2: Finding duplicate files between two folders.........................................................................................1101
   tFileList Standard properties............................................................................................................................................. 1097
   Scenario 1: Reading and extracting data from an XML structure............................................................................1092
   Scenario 2: Extracting erroneous XML data via a reject flow.............................................................................1093
   tFileInputXML Standard properties.................................................................................................................................1089
   Scenario 1: Reading and matching the keys and the values of different .properties files and outputting
   the results in a glossary............................................................................................................................................... 1096
   tFileInputRegex Standard properties..............................................................................................................................1084
   Scenario: Regex to Positional file................................................................................................................................... 1086
   tFileInputRaw................................................................................................................................................................. 1081
   Related Scenario........................................................................................................................................................... 1083
   Related Scenario..................................................................................................................................................................... 1082
   Related Scenario..................................................................................................................................................................... 1081
   Scenario: Reading and matching the keys and the values of different .properties files and outputting
   the results in a glossary............................................................................................................................................... 1080
   tFileOutputProperties Standard properties.................................................................................................................. 1075
   Related scenario..................................................................................................................................................................... 1074
   Related scenario..................................................................................................................................................................... 1073
   Related scenario..................................................................................................................................................................... 1072
<table>
<thead>
<tr>
<th>Task</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>tFirebirdOutput</td>
<td>1190</td>
</tr>
<tr>
<td>tFirebirdInput</td>
<td>1186</td>
</tr>
<tr>
<td>tFirebirdConnection</td>
<td>1184</td>
</tr>
<tr>
<td>tFirebirdCommit</td>
<td>1182</td>
</tr>
<tr>
<td>tFileUnarchive</td>
<td>1169</td>
</tr>
<tr>
<td>tFileProperties</td>
<td>1158</td>
</tr>
<tr>
<td>tFileRowCount</td>
<td>1161</td>
</tr>
<tr>
<td>tFileTouch</td>
<td>1167</td>
</tr>
<tr>
<td>tFileUnarchive</td>
<td>1169</td>
</tr>
<tr>
<td>tFilterColumns</td>
<td>1171</td>
</tr>
<tr>
<td>tFilterRow</td>
<td>1173</td>
</tr>
<tr>
<td>tFirebirdClose</td>
<td>1180</td>
</tr>
<tr>
<td>tFirebirdCommit</td>
<td>1182</td>
</tr>
<tr>
<td>tFirebirdConnection</td>
<td>1184</td>
</tr>
<tr>
<td>tFirebirdInput</td>
<td>1186</td>
</tr>
<tr>
<td>tFirebirdOutput</td>
<td>1190</td>
</tr>
</tbody>
</table>
Related scenarios..................................................................................................................................................................... 1195

**tFirebirdRollback** .................................................................................................................................................................. 1196
  tFirebirdRollback Standard properties................................................................................................................................. 1196
  Related scenario........................................................................................................................................................................ 1197

**tFirebirdRow** ........................................................................................................................................................................ 1198
  tFirebirdRow Standard properties.................................................................................................................................................. 1198
  Related scenarios........................................................................................................................................................................... 1201

**tFixedFlowInput** .................................................................................................................................................................... 1202
  tFixedFlowInput Standard properties.......................................................................................................................................... 1202
  Related scenarios........................................................................................................................................................................... 1203

**tFlowMeter** ........................................................................................................................................................................... 1204
  tFlowMeter Standard properties....................................................................................................................................................... 1204
  Related scenario........................................................................................................................................................................... 1205

**tFlowMeterCatcher** ................................................................................................................................................................. 1206
  tFlowMeterCatcher Standard properties..................................................................................................................................... 1206
  Scenario: Catching flow metrics from a Job.................................................................................................................................. 1207

**tFlowToIterate** .......................................................................................................................................................................... 1211
  tFlowToIterate Standard properties........................................................................................................................................... 1211
  Scenario: Transforming data flow to a list........................................................................................................................................ 1212

**tForeach** .................................................................................................................................................................................. 1216
  tForeach Standard properties......................................................................................................................................................... 1216
  Scenario: Iterating on a list and retrieving the values.................................................................................................................... 1217

**tFTPClose** ................................................................................................................................................................................ 1219
  tFTPClose Standard properties....................................................................................................................................................... 1219
  Related scenarios........................................................................................................................................................................... 1219

**tFTPConnection** .................................................................................................................................................................... 1220
  tFTPConnection Standard properties........................................................................................................................................... 1220
  Related scenarios........................................................................................................................................................................... 1220

**tFTPDelete** ............................................................................................................................................................................. 1223
  tFTPDelete Standard properties..................................................................................................................................................... 1223
  Related scenario........................................................................................................................................................................... 1223

**tFTPFileExist** ......................................................................................................................................................................... 1227
  tFTPFileExist Standard properties................................................................................................................................................ 1227
  Related scenario........................................................................................................................................................................... 1227

**tFTPFileList** ........................................................................................................................................................................... 1230
  tFTPFileList Standard properties................................................................................................................................................... 1230
  Listing and getting files/folders on an FTP directory................................................................................................................... 1232

**tFTPFileProperties** ................................................................................................................................................................. 1238
  tFTPFileProperties Standard properties................................................................................................................................. 1238
  Related scenario........................................................................................................................................................................... 1240

**tFTPGet** .................................................................................................................................................................................. 1241
  tFTPGet Standard properties......................................................................................................................................................... 1241
  Related scenario........................................................................................................................................................................... 1241

**tFTPPut** .................................................................................................................................................................................. 1245
  tFTPPut Standard properties......................................................................................................................................................... 1245
<table>
<thead>
<tr>
<th>Task Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>tGreenplumCommit</td>
<td>1324</td>
</tr>
<tr>
<td>tGreenplumCommit</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tGreenplumConnection</td>
<td>1326</td>
</tr>
<tr>
<td>tGreenplumConnection</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tGreenplumGLoad</td>
<td>1328</td>
</tr>
<tr>
<td>tGreenplumGLoad</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenario</td>
<td></td>
</tr>
<tr>
<td>tGreenplumInput</td>
<td>1334</td>
</tr>
<tr>
<td>tGreenplumInput</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tGreenplumOutput</td>
<td>1338</td>
</tr>
<tr>
<td>tGreenplumOutput</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tGreenplumOutputBulk</td>
<td>1344</td>
</tr>
<tr>
<td>tGreenplumOutputBulk</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tGreenplumOutputBulkExec</td>
<td>1347</td>
</tr>
<tr>
<td>tGreenplumOutputBulkExec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tGreenplumRollback</td>
<td>1351</td>
</tr>
<tr>
<td>tGreenplumRollback</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tGreenplumRow</td>
<td>1353</td>
</tr>
<tr>
<td>tGreenplumRow</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tGreenplumSCD</td>
<td>1358</td>
</tr>
<tr>
<td>tGreenplumSCD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tGroovy</td>
<td>1362</td>
</tr>
<tr>
<td>tGroovy</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tGroovyFile</td>
<td>1364</td>
</tr>
<tr>
<td>tGroovyFile</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Scenario: Calling a file which contains Groovy code</td>
<td>1365</td>
</tr>
<tr>
<td>tGSBucketCreate</td>
<td>1367</td>
</tr>
<tr>
<td>tGSBucketCreate</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenario</td>
<td></td>
</tr>
<tr>
<td>tGSBucketDelete</td>
<td>1369</td>
</tr>
<tr>
<td>tGSBucketDelete</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tGSBucketExist</td>
<td>1371</td>
</tr>
<tr>
<td>tGSBucketExist</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Related scenario</td>
<td></td>
</tr>
</tbody>
</table>
Scenario: Exchanging customer data with HBase

Scenario 1: Reading data from the cache memory for high-speed data access
Scenario 2: Clearing the memory before loading data to it in case an iterator exists in the same subjob.

Scenario: Managing files with Google Cloud Storage
Related scenarios

Scenario 2: Sending a POST request from a local JSON file

Scenario 1: Sending a HTTP request to the server and saving the response information to a local file.
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>tIngresSCD</td>
<td>1750</td>
</tr>
<tr>
<td>tInformixSCD</td>
<td>1750</td>
</tr>
<tr>
<td>tInformixSP</td>
<td>1754</td>
</tr>
<tr>
<td>tInformixBulkExec</td>
<td>1758</td>
</tr>
<tr>
<td>tInformixRow</td>
<td>1746</td>
</tr>
<tr>
<td>tInformixRow Standard properties</td>
<td>1746</td>
</tr>
<tr>
<td>Related scenario</td>
<td>1749</td>
</tr>
<tr>
<td>tIngresRow</td>
<td>1746</td>
</tr>
<tr>
<td>tIngresRow Standard properties</td>
<td>1746</td>
</tr>
<tr>
<td>Related scenarios</td>
<td>1749</td>
</tr>
<tr>
<td>tIngresClose</td>
<td>1762</td>
</tr>
<tr>
<td>tIngresClose Standard properties</td>
<td>1762</td>
</tr>
<tr>
<td>Related scenarios</td>
<td>1763</td>
</tr>
<tr>
<td>tIngresCommit</td>
<td>1764</td>
</tr>
<tr>
<td>tIngresCommit Standard properties</td>
<td>1764</td>
</tr>
<tr>
<td>Related scenario</td>
<td>1765</td>
</tr>
<tr>
<td>tIngresConnection</td>
<td>1766</td>
</tr>
<tr>
<td>tIngresConnection Standard properties</td>
<td>1766</td>
</tr>
<tr>
<td>Related scenarios</td>
<td>1767</td>
</tr>
<tr>
<td>tIngresInput</td>
<td>1768</td>
</tr>
<tr>
<td>tIngresInput Standard properties</td>
<td>1768</td>
</tr>
<tr>
<td>Related scenarios</td>
<td>1770</td>
</tr>
<tr>
<td>tIngresOutput</td>
<td>1771</td>
</tr>
<tr>
<td>tIngresOutput Standard properties</td>
<td>1771</td>
</tr>
<tr>
<td>Related scenarios</td>
<td>1775</td>
</tr>
<tr>
<td>tIngresOutputBulk</td>
<td>1776</td>
</tr>
<tr>
<td>tIngresOutputBulk Standard properties</td>
<td>1776</td>
</tr>
<tr>
<td>Related scenarios</td>
<td>1778</td>
</tr>
<tr>
<td>tIngresOutputBulkExec</td>
<td>1779</td>
</tr>
<tr>
<td>tIngresOutputBulkExec Standard properties</td>
<td>1779</td>
</tr>
<tr>
<td>Scenario: Loading data to a table in the Ingres DBMS</td>
<td>1782</td>
</tr>
<tr>
<td>Related scenarios</td>
<td>1785</td>
</tr>
<tr>
<td>tIngresRollback</td>
<td>1786</td>
</tr>
<tr>
<td>tIngresRollback Standard properties</td>
<td>1786</td>
</tr>
<tr>
<td>Related scenarios</td>
<td>1787</td>
</tr>
<tr>
<td>tIngresRow</td>
<td>1788</td>
</tr>
<tr>
<td>tIngresRow Standard properties</td>
<td>1788</td>
</tr>
<tr>
<td>Related scenarios</td>
<td>1791</td>
</tr>
<tr>
<td>tIngresSCD</td>
<td>1792</td>
</tr>
<tr>
<td>tIngresSCD Standard properties</td>
<td>1792</td>
</tr>
<tr>
<td>Related scenario</td>
<td>1795</td>
</tr>
</tbody>
</table>
Scenario: Transforming data line by line using tJavaRow................................................................. 1860

Scenario 2: Processing rows of data with tJavaFlex........................................................................... 1854

Scenario: Generating data flow ........................................................................................................ 1852

Scenario 2: Processing rows of data with tJavaFlex........................................................................... 1854

Scenario: Transforming data line by line using tJavaRow................................................................. 1860
Creating an issue in JIRA application................................................................. 1912
Updating an issue in JIRA application............................................................ 1916

tJMSInput............................................................................................................ 1920
tJMSInput Standard properties................................................................. 1920
Related scenarios......................................................................................... 1922

tJMSOutput........................................................................................................ 1923
tJMSOutput Standard properties............................................................... 1923
Scenario: Enqueuing/dequeuing a message on the ActiveMQ server .......... 1925
Related scenarios....................................................................................... 1927

tJoin..................................................................................................................... 1928
tJoin Standard properties............................................................... 1928
Scenario 1: Doing an exact match on two columns and outputting the main and rejected data................................. 1929

tKafkaCommit.................................................................................................... 1934
tKafkaCommit Standard properties....................................................... 1934
Related scenarios..................................................................................... 1934

tKafkaConnection............................................................................................ 1935
tKafkaConnection Standard properties............................................ 1935
Related scenarios..................................................................................... 1936

tKafkaCreateTopic.......................................................................................... 1937
tKafkaCreateTopic Standard properties........................................ 1937
Related scenarios..................................................................................... 1938

tKafkaInput.......................................................................................................... 1939
tKafkaInput Standard properties.......................................................... 1939
Related scenarios..................................................................................... 1943

tKafkaOutput...................................................................................................... 1944
tKafkaOutput Standard properties......................................................... 1944
Related scenarios..................................................................................... 1946

tLDAPAttributesInput..................................................................................... 1947
tLDAPAttributesInput Standard properties........................................ 1947
Related scenario....................................................................................... 1950

tLDAPClose....................................................................................................... 1951
tLDAPClose Standard properties.......................................................... 1951
Related scenarios..................................................................................... 1951

tLDAPConnection............................................................................................ 1952
tLDAPConnection Standard properties............................................ 1952
Related scenarios..................................................................................... 1953

tLDAPInput.......................................................................................................... 1954
tLDAPInput Standard properties.......................................................... 1954
Scenario: Displaying LDAP directory's filtered content.......................... 1957

tLDAPOutput...................................................................................................... 1959
tLDAPOutput Standard properties.......................................................... 1959
Scenario: Editing data in a LDAP directory............................................. 1962

tLDAPRenameEntry.......................................................................................... 1966
tLDAPRenameEntry Standard properties............................................... 1966
Related scenarios..................................................................................... 1968
Related scenarios

- tMapRStreamsConnection Standard properties
- Related scenario

- tMapRStreamsCommit Standard properties
- Related scenario

- tMapROjaiOutput Standard properties
- Related scenario

- tMapRDBOutput Standard properties
- Related scenario

- tMapRDBInput Standard properties
- Related scenario

- tMapRDBConnection Standard properties
- Related scenario

- tMapRDBClose
- Related scenario

- tMapStandard properties
- Related scenarios

- tLineChart Standard properties
- Related scenario

- tLogRow Standard properties
- Related scenario

- tLibraryLoad Standard properties
- Related scenario

- tLogCatcher Standard properties
- Related scenario

- tLogRow: Checking the format of an e-mail address

- tLogCatcher: Creating a line chart to ease trend analysis

- Advanced mapping with lookup reload at each row

- Advanced mapping with inner join rejections

- Advanced mapping using filters, explicit joins, and rejections

- Related scenario

- Caching messages triggered by a tWarn component
- Related scenario

- Caching the message triggered by a tDie component

- Related scenario

- Scenario: Checking the format of an e-mail address

- Scenario: Job execution in a loop

- Mapping data using a filter and a simple explicit join

- Mapping data using inner join rejections

- Related scenario

- Writing candidate data in a MapR-DB OJAI database

- Related scenarios

- Related scenarios

- Related scenarios

- Related scenarios
Data transmission with Marketo using REST API

Adding a lead record to a Marketo list using SOAP API
Related scenario: Using MongoDB functions to create a collection and write data to it........................................... 2376

Scenario 2: Upserting records in a collection .............................................................................................................2365
Scenario 1: Creating a collection and writing data to it ........................................................................................ 2361

Related scenario................................................................................................................................................................... 2355
Related scenario..................................................................................................................................................................... 2336
Related scenario..................................................................................................................................................................... 2332
Related scenario....................................................................................................................................................................2317
Related scenario..................................................................................................................................................................... 2300
Related scenario..................................................................................................................................................................... 2292
Related scenario..................................................................................................................................................................... 2293
Related scenario..................................................................................................................................................................... 2298
Related scenario..................................................................................................................................................................... 2299
Related scenario..................................................................................................................................................................... 2300
Related scenario..................................................................................................................................................................... 2301
Related scenario..................................................................................................................................................................... 2303
Related scenario..................................................................................................................................................................... 2309
Related scenario..................................................................................................................................................................... 2317
Related scenario..................................................................................................................................................................... 2317
Related scenario..................................................................................................................................................................... 2318
Related scenario..................................................................................................................................................................... 2320
Related scenario..................................................................................................................................................................... 2321
Related scenario..................................................................................................................................................................... 2324
Related scenario..................................................................................................................................................................... 2325
Related scenario..................................................................................................................................................................... 2328
Related scenario..................................................................................................................................................................... 2329
Related scenario..................................................................................................................................................................... 2332
Related scenario..................................................................................................................................................................... 2333
Related scenario..................................................................................................................................................................... 2336
Related scenario..................................................................................................................................................................... 2337
Related scenarios....................................................................................................................................................................2340
Scenario: Retrieving data from a collection by advanced queries.......................................................... 2352
Scenario 1: Creating a collection and writing data to it......................................................................................... 2361
Scenario 2: Upserting records in a collection........................................................................................................... 2365
Scenario: Using MongoDB functions to create a collection and write data to it.............................................. 2373
Scenario: Inserting data into a database table and extracting useful information from it

Scenario 2: Deleting data from a defined table in a MicrosoftAX server

Scenario 1: Inserting data in a defined table in a MicrosoftAX server

Scenario: ‘Hello world!’ type test
Related scenarios..................................................................................................................................................................... 2444
tMSSQLSCD......................................................................................................................................................................................... 2445
tMSSQLSCD Standard properties................................................................................................................................................ 2445
Related scenario........................................................................................................................................................................ 2448
tMSSQLSP........................................................................................................................................................................................ 2449
tMSSQLSP Standard properties................................................................................................................................................ 2449
Scenario: Retrieving personal information using a stored procedure....................................................................................2452
Related scenarios........................................................................................................................................................................ 2458
tMSSQLTableList.................................................................................................................................................................................. 2459
tMSSQLTableList Standard properties........................................................................................................................................ 2459
Related scenario........................................................................................................................................................................ 2460
tMysqlBulkExec.................................................................................................................................................................................. 2461
tMysqlBulkExec Standard properties........................................................................................................................................ 2461
Related scenarios........................................................................................................................................................................ 2464
tMysqlClose......................................................................................................................................................................................... 2466
tMysqlClose Standard properties................................................................................................................................................ 2466
Related scenario........................................................................................................................................................................ 2467
tMysqlColumnList............................................................................................................................................................................. 2468
tMysqlColumnList Standard properties........................................................................................................................................ 2468
Scenario: Iterating on a DB table and listing its column names.................................................................................................2470
tMysqlCommit.................................................................................................................................................................................. 2473
tMysqlCommit Standard properties........................................................................................................................................ 2473
Related scenario........................................................................................................................................................................ 2474
tMysqlConnection............................................................................................................................................................................. 2475
tMysqlConnection Standard properties........................................................................................................................................ 2475
Scenario: Inserting data in mother/daughter tables................................................................................................................2476
Scenario: Sharing a database connection between a parent Job and child Job........................................................................2480
tMysqlInput...................................................................................................................................................................................... 2488
tMysqlInput Standard properties................................................................................................................................................ 2488
Scenario: Writing columns from a MySQL database to an output file using tMysqlInput.......................................................2492
Scenario: Using context parameters when reading a table from a database...........................................................................2494
Scenario: Reading data from databases through context-based dynamic connections.........................................................2497
Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database...................................................2503
tMysqlLastInsertId........................................................................................................................................................................... 2508
tMysqlLastInsertId Standard properties..................................................................................................................................... 2508
Scenario: Get the ID for the last inserted record with tMysqlLastInsertId............................................................................2510
tMysqlLookupInput.......................................................................................................................................................................... 2514
tMysqlOutput................................................................................................................................................................................... 2515
tMysqlOutput Standard properties................................................................................................................................................ 2515
Scenario: Inserting a column and altering data using tMysqlOutput.........................................................................................2521
Scenario: Updating data using tMysqlOutput............................................................................................................................2526
Scenario: Retrieving data in error with a Reject link................................................................................................................2529
tMysqlOutputBulk............................................................................................................................................................................. 2535
tMysqlOutputBulk Standard properties...................................................................................................................................... 2535
Scenario: Inserting transformed data in MySQL database....................................................................................................2537
Scenario: Iterating on DB tables and deleting their content using a user-defined SQL template

Scenario: Using PreparedStatement objects to query data

Tracking data changes using Slowly Changing Dimensions (type 0 through type 3)

Scenario: Combining two flows for selective output

Scenario: Inserting data in bulk in MySQL database

Related Scenario

Related scenarios...
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>tNetezzaRollback</td>
<td>2701</td>
</tr>
<tr>
<td>tNetezzaOutput</td>
<td>2695</td>
</tr>
<tr>
<td>tNetezzaNzLoad</td>
<td>2688</td>
</tr>
<tr>
<td>tNetezzaInput</td>
<td>2684</td>
</tr>
<tr>
<td>tNetezzaConnection</td>
<td>2682</td>
</tr>
<tr>
<td>tNetezzaCommit</td>
<td>2680</td>
</tr>
<tr>
<td>tNetezzaBulkExec</td>
<td>2674</td>
</tr>
<tr>
<td>tNetezzaClose</td>
<td>2678</td>
</tr>
<tr>
<td>tNetezzaOutputRelationship</td>
<td>2645</td>
</tr>
<tr>
<td>tNeo4jRow</td>
<td>2657</td>
</tr>
<tr>
<td>tNeo4jInput</td>
<td>2628</td>
</tr>
<tr>
<td>tNeo4jImportTool</td>
<td>2626</td>
</tr>
<tr>
<td>tNeo4jConnection</td>
<td>2624</td>
</tr>
<tr>
<td>tNeo4jOutputRelationship</td>
<td>2645</td>
</tr>
<tr>
<td>tNeo4jOutputRelationship</td>
<td>2645</td>
</tr>
<tr>
<td>tNeo4jRow</td>
<td>2657</td>
</tr>
<tr>
<td>tNeo4jInput</td>
<td>2628</td>
</tr>
<tr>
<td>tNeo4jImportTool</td>
<td>2626</td>
</tr>
<tr>
<td>tNeo4jConnection</td>
<td>2624</td>
</tr>
<tr>
<td>tNeo4jOutput</td>
<td>2631</td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>Scenario: Writing data to a Neo4j database and reading specific data from it</td>
<td>2635</td>
</tr>
<tr>
<td>Scenario 2: Writing family information to Neo4j and creating relationships</td>
<td>2639</td>
</tr>
<tr>
<td>tNeo4jOutputRelationship</td>
<td>2645</td>
</tr>
<tr>
<td>Scenario: Writing information of actors and movies to Neo4j with hierarchical relationship</td>
<td>2648</td>
</tr>
<tr>
<td>tNeo4jRow</td>
<td>2657</td>
</tr>
<tr>
<td>Scenario 1: Creating nodes with a label using a Cypher query</td>
<td>2660</td>
</tr>
<tr>
<td>Scenario 2: Importing data from a CSV file to Neo4j using a Cypher query</td>
<td>2664</td>
</tr>
<tr>
<td>Scenario 3: Importing data from a CSV file to Neo4j and creating relationships using a single Cypher query</td>
<td>2670</td>
</tr>
<tr>
<td>tNetezzaBulkExec</td>
<td>2674</td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>Scenario: Writing information of actors and movies to Neo4j with hierarchical relationship</td>
<td>2648</td>
</tr>
<tr>
<td>tNetezzaClose</td>
<td>2678</td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tNetezzaCommit</td>
<td>2680</td>
</tr>
<tr>
<td>Related scenario</td>
<td></td>
</tr>
<tr>
<td>tNetezzaConnection</td>
<td>2682</td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tNetezzaBulkExec</td>
<td>2674</td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>Scenario: Writing data to a Neo4j database and reading specific data from it</td>
<td>2635</td>
</tr>
<tr>
<td>Scenario 2: Writing family information to Neo4j and creating relationships</td>
<td>2639</td>
</tr>
<tr>
<td>tNetezzaClose</td>
<td>2678</td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
<tr>
<td>tNetezzaCommit</td>
<td>2680</td>
</tr>
<tr>
<td>Related scenario</td>
<td></td>
</tr>
<tr>
<td>tNetezzaConnection</td>
<td>2682</td>
</tr>
<tr>
<td>Related scenarios</td>
<td></td>
</tr>
</tbody>
</table>
Discovering the read-only output schema of tPaloCubeList

Scenario: Creating a cube in an existing database

Related scenario:

Scenario: Checking number format using a stored procedure

Related scenario:

Related scenarios:

Related scenarios:

Related scenarios:
tPaloCubeList Standard properties .................................................................................................................... 2827
Scenario: Retrieving detailed cube information from a given database ................................................................. 2829

**tPaloDatabase** ..................................................................................................................................................... 2831
  tPaloDatabase Standard properties .................................................................................................................... 2831
  Scenario: Creating a database ............................................................................................................................ 2832

**tPaloDatabaseList** ............................................................................................................................................... 2834
  Discovering the read-only output schema of tPaloDatabaseList ........................................................................ 2834
  tPaloDatabaseList Standard properties ........................................................................................................... 2834
  Scenario: Retrieving detailed database information from a given Palo server ................................................ 2836

**tPaloDimension** .................................................................................................................................................. 2838
  tPaloDimension Standard properties .................................................................................................................. 2838
  Scenario: Creating a dimension with elements ................................................................................................ 2841

**tPaloDimensionList** ............................................................................................................................................ 2846
  Discovering the read-only output schema of tPaloDimensionList ..................................................................... 2846
  tPaloDimensionList Standard properties .......................................................................................................... 2846
  Scenario: Retrieving detailed dimension information from a given database .................................................. 2849

**tPaloInputMulti** .................................................................................................................................................. 2851
  tPaloInputMulti Standard properties .................................................................................................................. 2851
  Scenario: Retrieving dimension elements from a given cube ............................................................................. 2853

**tPaloOutput** .......................................................................................................................................................... 2857
  tPaloOutput Standard properties ......................................................................................................................... 2857
  Related scenario .................................................................................................................................................. 2859

**tPaloOutputMulti** ................................................................................................................................................. 2860
  tPaloOutputMulti Standard properties ............................................................................................................... 2860
  Scenario 1: Writing data into a given cube ......................................................................................................... 2862
  Scenario 2: Rejecting inflow data when the elements to be written do not exist in a given cube ..................... 2865

**tPaloRule** .............................................................................................................................................................. 2870
  tPaloRule Standard properties ............................................................................................................................ 2870
  Scenario: Creating a rule in a given cube ............................................................................................................. 2872

**tPaloRuleList** ..................................................................................................................................................... 2874
  Discovering the read-only output schema of tPaloRuleList .............................................................................. 2874
  tPaloRuleList Standard properties .................................................................................................................... 2874
  Scenario: Retrieving detailed rule information from a given cube ................................................................... 2876

**tParAccelBulkExec** ............................................................................................................................................. 2879
  tParAccelBulkExec Standard properties ............................................................................................................ 2879
  Related scenarios ................................................................................................................................................. 2883

**tParAccelClose** .................................................................................................................................................... 2884
  tParAccelClose Standard properties .................................................................................................................... 2884
  Related scenarios ................................................................................................................................................. 2885

**tParAccelCommit** ................................................................................................................................................. 2886
  tParAccelCommit Standard properties .................................................................................................................. 2886
  Related scenario .................................................................................................................................................. 2887

**tParAccelConnection** ......................................................................................................................................... 2888
  tParAccelConnection Standard properties .......................................................................................................... 2888
  Related scenario .................................................................................................................................................. 2889
Scenario: Filtering rows of data based on a condition and saving the result to a local file

`tPigFilterRow` Standard properties .......................................................... 2948
Related scenarios ..................................................................................... 2893

`tPigFilterColumns`................................................................................. 2945
`tPigDistinct`............................................................................................ 2942
Related scenarios ..................................................................................... 2899

`tPigCross`............................................................................................... 2939
Related scenarios ..................................................................................... 2941

Scenario: aggregating data from two relations using COGROUP
`tPigCoGroup` Standard properties ............................................................ 2929
Related Scenario ...................................................................................... 2919

`tPigAggregate`.......................................................................................... 2920
Related scenario ....................................................................................... 2922

`tPigCode`................................................................................................... 2923
Scenario: Selecting a column of data from an input file and store it into a local file... 2925

`tPigCoGroup`............................................................................................. 2929
Scenario: aggregating data from two relations using COGROUP ................. 2931

`tPigCross`............................................................................................... 2939
Related scenarios ..................................................................................... 2941

`tPigDistinct`............................................................................................. 2942
Related scenario ....................................................................................... 2944

`tPigFilterColumns`.................................................................................. 2945
Related Scenario ...................................................................................... 2947

`tPigFilterRow`........................................................................................ 2948
Scenario: Filtering rows of data based on a condition and saving the result to a local file... 2950

Related scenarios ..................................................................................... 2899
tPigJoin .................................................................................................................................................. 2954
  tPigJoin Standard properties .................................................................................................................. 2954
  Scenario: Joining two files based on an exact match and saving the result to a local file ..................... 2957

TPigLoad .................................................................................................................................................. 2964
  tPigLoad Standard properties ................................................................................................................. 2964
  Scenario: Loading an HBase table .......................................................................................................... 2978

TPigMap .................................................................................................................................................. 2982
  Optional map settings .............................................................................................................................. 2982
  tPigMap Standard properties .................................................................................................................. 2983
  Scenario: Joining data about road conditions in a Pig process ................................................................ 2984

TPigReplicate .......................................................................................................................................... 2992
  tPigReplicate Standard properties ......................................................................................................... 2992
  Scenario: Replicating a flow and sorting two identical flows respectively ............................................. 2994

TPigSort .................................................................................................................................................. 3001
  tPigSort Standard properties ................................................................................................................... 3001
  Scenario: Sorting data in ascending order .............................................................................................. 3003

TPigStoreResult ....................................................................................................................................... 3007
  tPigStoreResult Standard properties ...................................................................................................... 3007
  Related Scenario ..................................................................................................................................... 3013

TPivotToColumnsDelimited ...................................................................................................................... 3014
  tPivotToColumnsDelimited Standard properties .................................................................................. 3014
  Scenario: Using a pivot column to aggregate data .................................................................................... 3015

TPop .......................................................................................................................................................... 3018
  tPOP Standard properties ....................................................................................................................... 3018
  Scenario: Retrieving a selection of email messages from an email server ............................................ 3020

tPostgresPlusBulkExec ............................................................................................................................. 3023
  tPostgresPlusBulkExec Standard properties ......................................................................................... 3023
  Related scenarios ...................................................................................................................................... 3026

tPostgresPlusClose .................................................................................................................................... 3027
  tPostgresPlusClose Standard properties ............................................................................................... 3027
  Related scenarios .................................................................................................................................... 3028

tPostgresPlusCommit ................................................................................................................................. 3029
  tPostgresPlusCommit Standard properties ............................................................................................. 3029
  Related scenario ....................................................................................................................................... 3030

tPostgresPlusConnection ........................................................................................................................... 3031
  tPostgresPlusConnection Standard properties ....................................................................................... 3031
  Related scenario ....................................................................................................................................... 3032

tPostgresPlusInput ..................................................................................................................................... 3033
  tPostgresPlusInput Standard properties ............................................................................................... 3033
  Related scenarios ....................................................................................................................................... 3036

tPostgresPlusOutput ................................................................................................................................. 3037
  tPostgresPlusOutput Standard properties .............................................................................................. 3037
  Related scenarios ....................................................................................................................................... 3042

tPostgresPlusOutputBulk ........................................................................................................................... 3043
  tPostgresPlusOutputBulk Standard properties ....................................................................................... 3043
Scenario 1: Using URI Query parameters to explore the data of a database

Scenario 2: Updating user information by interacting with a RESTful service

Scenario 3: REST service accepting HTTP POST requests

Related scenarios:
- tRiakKeyList Standard properties
- tRiakInput Standard properties
- tRiakClose Standard properties
- tRiakBucketList Standard properties
- tRESTResponse Standard properties
- tRESTRequest Standard properties
- tRESTClient Standard properties
- tReplaceList Standard properties
- tReplace Standard properties
- tRedshiftUnload Standard properties

Related Scenario:
- Scenario: Replacement from a reference file
- Scenario: Replicating a flow and sorting two identical flows respectively
- Scenario: Multiple replacements and column filtering
- Scenario: Creating and retrieving data by invoking REST Web service
- Scenario 1: Getting user information by interacting with a RESTful service
- Scenario 2: Updating user information by interacting with a RESTful service
- Scenario: Creating and retrieving data by invoking REST Web service
- Scenario: Replicating a flow and sorting two identical flows respectively
- Scenario 1: Getting user information by interacting with a RESTful service
- Scenario 2: Updating user information by interacting with a RESTful service
- Scenario 1: Multiple replacements and column filtering
Scenario: Verifying the absence of a bucket, creating it and listing all the S3 buckets.

tS3BucketExist Standard properties
Scenario: Verifying the absence of a bucket, creating it and listing all the S3 buckets.

Propagating the buffered output data from the child Job to the parent Job.

Running a list of child Jobs dynamically.

Calling a Job and passing the parameter needed to the called Job.

Related scenario: Exchanging messages between a Job and a Route.

tRunJob Standard properties
Calling a Job and passing the parameter needed to the called Job.
Running a list of child Jobs dynamically.
Propagating the buffered output data from the child Job to the parent Job.

tRSSInput Standard properties
Scenario: Fetching frequently updated blog entries.

Scenario 1: Creating an RSS flow and storing files on an FTP server.
Scenario 2: Creating an RSS flow that contains metadata.
Scenario 3: Creating an ATOM feed XML file.

Scenario: Exchanging messages between a Job and a Route.

Related scenarios:

Scenario: Exchanging messages between a Job and a Route.

tRiakOutput Standard properties
Related scenarios:

Scenario: Exchanging messages between a Job and a Route.

Scenario: Verifying the absence of a bucket, creating it and listing all the S3 buckets.

Scenario 1: Creating an RSS flow and storing files on an FTP server.
Scenario 2: Creating an RSS flow that contains metadata.
Scenario 3: Creating an ATOM feed XML file.

Scenario: Exchanging messages between a Job and a Route.

Related scenarios:

Scenario: Exchanging messages between a Job and a Route.

Related scenarios:

Scenario: Exchanging messages between a Job and a Route.
Upserting Salesforce data based on external IDs..................................................................................................... 3420
How to set schema for the guess query feature of tSalesforceInput ..............................................................................3410
Related scenario..................................................................................................................................................................... 3403
Recovering deleted data from Salesforce .....................................................................................................................3390
Related scenario..................................................................................................................................................................... 3385
Connecting to Salesforce using OAuth implicit flow to authenticate the user (deprecated)............................3381
Related scenario..................................................................................................................................................................... 3377
Related scenario..................................................................................................................................................................... 3377
Recovering deleted data from Salesforce.....................................................................................................................3386
Related scenario..................................................................................................................................................................... 3385
Scenario: Using a Sage X3 Web service to insert data into a given Sage X3 system..............................................3369
Related scenario..................................................................................................................................................................... 3361
Recovering deleted data from Salesforce.....................................................................................................................3386
Related scenario..................................................................................................................................................................... 3385
Scenario: Using a Sage X3 Web service to insert data into a given Sage X3 system..............................................3369
Relatied scenario..................................................................................................................................................................... 3363
Related scenario..................................................................................................................................................................... 3361
Related scenario: Filtering rows and groups of rows.

Scenario: Filtering rows and groups of rows.

Related scenario:

Related scenario:

tSalesforceOutputBulk Exec Standard properties

Scenario: Inserting bulk data into Salesforce.

TSalesforceOutputBulk Exec Standard properties

Related scenario:

Related scenario:

tSalesforceWaveBulk Exec Standard properties

Scenario: Inserting bulk data into Salesforce.

tSalesforceWaveBulk Exec Standard properties

Related scenario:

Related scenario:

tSalesforceWaveOutputBulk Exec Standard properties

Scenario: Inserting bulk data into Salesforce.

tSalesforceWaveOutputBulk Exec Standard properties

Related scenario:

Related scenario:

tSampleRow Standard properties

Scenario: Filtering rows and groups of rows.

tSampleRow Standard properties

Related scenario:

Related scenario:

tSAP Hana Close Standard properties

Scenario: Filtering rows and groups of rows.

tSAP Hana Close Standard properties

Related scenarios:

Related scenarios:

tSAP Hana Commit Standard properties

Scenario: Filtering rows and groups of rows.

tSAP Hana Commit Standard properties

Related scenario:

Related scenario:

tSAP Hana Connection Standard properties

Scenario: Filtering rows and groups of rows.

tSAP Hana Connection Standard properties

Related scenarios:

Related scenarios:

tSAP Hana Input Standard properties

Scenario: Filtering rows and groups of rows.

tSAP Hana Input Standard properties

Related scenarios:

Related scenarios:

tSAP Hana Output Standard properties

Scenario: Filtering rows and groups of rows.

tSAP Hana Output Standard properties

Related scenarios:

Related scenarios:

tSAP Hana Rollback Standard properties

Scenario: Filtering rows and groups of rows.

tSAP Hana Rollback Standard properties

Related scenarios:

Related scenarios:

tSAP Hana Row Standard properties

Scenario: Filtering rows and groups of rows.

tSAP Hana Row Standard properties

Related scenarios:

Related scenarios:

tSas Input Standard properties

Scenario: Filtering rows and groups of rows.

tSas Input Standard properties

Related scenarios:

Related scenarios:

tSas Output Standard properties

Scenario: Filtering rows and groups of rows.

tSas Output Standard properties

Related scenarios:

Related scenarios:

tSchema Compliance Check Standard properties

Scenario: Filtering rows and groups of rows.

tSchema Compliance Check Standard properties

Related scenario:

Related scenario:

tSCPClose Standard properties

Scenario: Filtering rows and groups of rows.

tSCPClose Standard properties

Related scenario:

Related scenario:
Scenario: Printing out the content of a global variable

tSetGlobalVar

tSetGlobalVar Standard properties
Related scenario

Scenario: Validating the status of the connection to a remote host
tServerAlive

tServerAlive Standard properties
Scenario: Validating the status of the connection to a remote host

Scenario: Email on error
tSendMail

tSendMail Standard properties
Scenario: Email on error

Scenario: Email on error
tServerAlive

tServerAlive Standard properties
Scenario: Validating the status of the connection to a remote host

Scenario 1: Splitting one row into two rows

Scenario: Passing on data to the listening port

Related scenario

Related scenarios

Related scenario

Related scenario

Related scenarios

Scenario: Extracting customer information from a private WSDL file

Related scenario

Related scenarios

Related scenarios
tSqoopImportAllTables Standard properties................................................................. 3711
Related scenarios................................................................................................. 3639

tSqoopImport Standard properties................................................................. 3695
Scenario: Importing a MySQL table to HDFS...................................................... 3705
Related scenarios................................................................................................. 3695

tSqoopExport Standard properties................................................................. 3684
Additional arguments......................................................................................... 3684
Related scenarios................................................................................................. 3685

t SqoopImportAllTables Standard properties.................................................. 3711
Related scenarios................................................................................................. 3711
Related scenarios................................................................................................................................. 3719

**tSqoopMerge**................................................................................................................................. 3720
  tSqoopMerge Standard properties................................................................................................. 3720
  Scenario: Merging two datasets in HDFS.................................................................................... 3728

**tSQSConnection**............................................................................................................................ 3733
  tSQSConnection Standard properties.......................................................................................... 3733
  Related scenarios..................................................................................................................... 3734

**tSQSInput**...................................................................................................................................... 3735
  tSQSInput Standard properties.................................................................................................... 3735
  Retrieving messages from an Amazon SQS queue...................................................................... 3737

**tSQSMessageChangeVisibility**.................................................................................................. 3743
  tSQSMessageChangeVisibility Standard properties.................................................................... 3743
  Related scenario....................................................................................................................... 3745

**tSQSMessageDelete**.................................................................................................................... 3746
  tSQSMessageDelete Standard properties.................................................................................... 3746
  Related scenario....................................................................................................................... 3747

**tSQSOutput**.................................................................................................................................... 3748
  tSQSOutput Standard properties................................................................................................. 3748
  Delivering messages to an Amazon SQS queue.......................................................................... 3750

**tSQSQueueAttributes**.................................................................................................................. 3757
  tSQSQueueAttributes Standard properties................................................................................ 3757
  Related scenario....................................................................................................................... 3759

**tSQSQueueCreate**......................................................................................................................... 3760
  tSQSQueueCreate Standard properties....................................................................................... 3760
  Related scenario....................................................................................................................... 3762

**tSQSQueueDelete**.......................................................................................................................... 3763
  tSQSQueueDelete Standard properties......................................................................................... 3763
  Related scenario....................................................................................................................... 3764

**tSQSQueueList**............................................................................................................................. 3765
  tSQSQueueList Standard properties.......................................................................................... 3765
  Listing Amazon SQS queues in an AWS region........................................................................ 3766

**tSQSQueuePurge**........................................................................................................................... 3771
  tSQSQueuePurge Standard properties........................................................................................ 3771
  Related scenario....................................................................................................................... 3772

**tSSH**............................................................................................................................................ 3773
  tSSH Standard properties.......................................................................................................... 3773
  Scenario: Remote system information display via SSH............................................................ 3776

**tStatCatcher**................................................................................................................................ 3778
  tStatCatcher Standard properties............................................................................................. 3778
  Scenario: Displaying the statistics log of Job execution............................................................ 3779

**tSugarCRMInput**.......................................................................................................................... 3783
  tSugarCRMInput Standard properties......................................................................................... 3783
  Scenario: Extracting account data from SugarCRM................................................................. 3784

**tSugarCRMOutput**....................................................................................................................... 3787
  tSugarCRMOutput Standard properties..................................................................................... 3787
Scenario 1: Waiting for a file to be created and stopping the iteration loop after a message is triggered.

Scenario 2: Waiting for a file to be created and continuing the iteration loop after a message is triggered.
Scenario: Getting country names through a Web service

Configuring a JSON Tree

Related Scenarios

Configuring a JSON Tree

Related Scenarios

Scenario: Writing flat data into JSON fields

Related Scenarios

Configuring a JSON Tree

Related Scenarios

Scenario: Extracting the structure of an XML file and inserting it into the fields of a database table

Related Scenarios

Configuring a JSON Tree

Related Scenarios

Scenario: Guessing the State name from an XMLRPC

Related Scenarios

Configuring a JSON Tree

Related Scenarios

Scenario: Validating data flows against an XSD file

Related Scenarios

Configuring a JSON Tree

Related Scenarios

Scenario: Transforming XML to html using an XSL stylesheet
atinject, dropbox-sdk-java: Java library for the Dropbox Core API, google-guice. Licensed under their respective license.
tAccessBulkExec

Offers gains in performance when carrying out Insert operations in an Access database.

The tAccessOutputBulk and tAccessBulkExec components are generally used together to output data to a delimited file and then to perform various actions on the file in an Access database, in a two step process. These two steps are fused together in the tAccessOutputBulkExec component, detailed in a separate section. The advantage of using a two step process is that it makes it possible to carry out transformations on the data before loading it in the database.

This component executes an Insert action on the data provided.

tAccessBulkExec Standard properties

These properties are used to configure tAccessBulkExec running in the Standard Job framework.

The Standard tAccessBulkExec component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data is stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
</tr>
<tr>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
<td></td>
</tr>
<tr>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
<td></td>
</tr>
</tbody>
</table>
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB version</td>
<td>Select the version of your database.</td>
</tr>
<tr>
<td>Database</td>
<td>Type in the directory where your database is stored.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations: <strong>None</strong>: No operation is carried out. <strong>Drop and create table</strong>: The table is removed and created again. <strong>Create table</strong>: The table does not exist and gets created. <strong>Create table if not exists</strong>: The table is created if it does not exist. <strong>Drop table if exists and create</strong>: The table is removed if it already exists and created again. <strong>Clear table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time and that the table must exist already for the insert operation to succeed.</td>
</tr>
<tr>
<td>Local filename</td>
<td>Browse to the delimited file to be loaded into your database.</td>
</tr>
<tr>
<td>Action on data</td>
<td>On the data of the table defined, you can perform: <strong>Insert</strong>: Add new entries to the table.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. <strong>Built-in</strong>: The schema is created and stored locally for this component only. Related topic: see <em>Talend Studio User Guide</em>. <strong>Repository</strong>: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Advanced settings

<table>
<thead>
<tr>
<th>Additional JDBC parameters</th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <strong>Use an existing connection</strong> check box in the Basic settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Include header</td>
<td>Select this check box to include the column header.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with <strong>tAccessOutputBulk</strong> component. Used together, they can offer gains in performance while feeding an Access database.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Dynamic settings</th>
<th>Click the [+] button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The <strong>Dynamic settings</strong> table is available only when the <strong>Use an existing connection</strong> check box is selected in the <strong>Basic settings</strong> view. Once a dynamic parameter is defined, the <strong>Component List</strong> box in the <strong>Basic settings</strong> view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>If you are using an ODBC driver, make sure that your JVM and ODBC versions match up: both 64-bit or 32-bit.</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

## Related scenarios

For use cases in relation with **tAccessBulkExec**, see the following scenarios:

- Scenario: Inserting transformed data in MySQL database on page 2537
- Scenario: Inserting data in bulk in MySQL database on page 2544
tAccessClose

Closes an active connection to the Access database so as to release occupied resources.

**tAccessClose Standard properties**

These properties are used to configure tAccessClose running in the Standard Job framework.

The Standard tAccessClose component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tAccessConnection component in the list if more than one connection is planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Usage**

| Usage rule | This component is to be used along with other Access components, especially with tAccessConnection and tAccessCommit. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context- |

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-
based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

| Limitation       | If you are using an ODBC driver, you need to work with Java 7, and make sure that your JVM and ODBC versions match up: both 64-bit or 32-bit. |

**Related scenarios**

No scenario is available for the Standard version of this component yet.
**tAccessCommit**

Commits in one go a global transaction instead of doing that on every row or every batch, and provides gain in performance, using a unique connection.

tAccessCommit validates the data processed through the Job into the connected database.

**tAccessCommit Standard properties**

These properties are used to configure tAccessCommit running in the Standard Job framework.

The Standard tAccessCommit component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tAccessConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task. <strong>Warning:</strong> If you want to use a Row &gt; Main connection to link tAccessCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tAccess* components, especially with the tAccessConnection and tAccessRollback components. |
| Dynamic settings | Click the \([+]\) button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |
| Limitation | If you are using an ODBC driver, you need to work with Java 7, and make sure that your JVM and ODBC versions match up: both 64-bit or 32-bit. |

## Related scenario

For **tAccessCommit** related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476
tAccessConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.
tAccessConnection opens a connection to the database for a current transaction.

tAccessConnection Standard properties

These properties are used to configure tAccessConnection running in the Standard Job framework.
The Standard tAccessConnection component belongs to the Databases and the ELT families.
The component in this framework is generally available.

ℹ️ Note:
This component is a specific version of a dynamic database connector. The properties related
to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components
on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Access 2003 or later versions.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td>Use or register a shared DB Connection</td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the <strong>Shared DB Connection Name</strong> field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child. This option is incompatible with the <strong>Use dynamic job</strong> and <strong>Use an independent process to run subjob</strong> options</td>
</tr>
</tbody>
</table>
of the tRunJob component. Using a shared connection together with a tRunJob component with either of these two options enabled will cause your Job to fail.

### Advanced settings

<table>
<thead>
<tr>
<th>Advanced settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional JDBC parameters</strong></td>
<td>Specify additional connection properties for the DB connection you are creating.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage rule</strong></td>
<td>This component is more commonly used with other tAccess* components, especially with the tAccessCommit and tAccessRollback components.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Limitation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Limitation</strong></td>
<td>If you are using an ODBC driver, you need to work with Java 7, and make sure that your JVM and ODBC versions match up: both 64-bit or 32-bit. When working with Java 8, this component supports only the General collation mode of Access.</td>
</tr>
</tbody>
</table>

### Scenario: Inserting data in parent/child tables

The following Job is dedicated to advanced database users, who want to carry out multiple table insertions using a parent table **Table1** to generate two child tables: **Name** and **Birthday**.

- In Access 2007, create an Access database named **Database1**.
- Once the Access database is created, create a table named **Table1** with two column headings: **Name** and **Birthday**.

Back into the Integration perspective of Talend Studio, the Job requires twelve components including tAccessConnection, tAccessCommit, tAccessInput, tAccessOutput and tAccessClose.
- Drop the following components from the Palette to the design workspace: tFileList, tFileInputDelimited, tMap, tAccessOutput (two), tAccessInput (two), tAccessCommit, tAccessClose and tLogRow (x2).

- Connect the tFileList component to the input file component using an Iterate link. Thus, the name of the file to be processed will be dynamically filled in from the tFileList directory using a global variable.

- Connect the tFileInputDelimited component to the tMap component and dispatch the flow between the two output Access components. Use a Row link for each of these connections representing the main data flow.

- Set the tFileList component properties, such as the directory where files will be fetched from.

- Add a tAccessConnection component and connect it to the starter component of this Job. In this example, the tFileList component uses an OnComponentOk link to define the execution order.

- In the tAccessConnection Component view, set the connection details manually or fetch them from the Repository if you centrally store them as a Metadata DB connection entry. For more information about Metadata, see Talend Studio User Guide.

- In the tFileInputDelimited component’s Basic settings view, press Ctrl+Space bar to access the variable list. Set the File Name field to the global variable: tFileList_1.CURRENT_FILEPATH. For more information about using variables, see Talend Studio User Guide.
• Set the rest of the fields as usual, defining the row and field separators according to your file structure.

• Then set the schema manually through the [Edit schema] dialog box or select the schema from the Repository. Make sure the data type is correctly set, in accordance with the nature of the data processed.

• In the tMap Output area, add two output tables, one called Name for the Name table, the second called Birthday, for the Birthday table. For more information about the tMap component, see Talend Studio User Guide.

• Drag the Name column from the Input area, and drop it to the Name table.

• Drag the Birthday column from the Input area, and drop it to the Birthday table.

• Then connect the output row links to distribute the flow correctly to the relevant DB output components.

• In each of the tAccessOutput components’ Basic settings view, select the Use an existing connection check box to retrieve the tAccessConnection details.
• Set the **Table** name making sure it corresponds to the correct table, in this example either *Name* or *Birthday*.

• There is no action on the table as they are already created.

• Select **Insert** as **Action on data** for both output components.

• Click on **Sync columns** to retrieve the schema set in the **tMap**.

• Then connect the first **tAccessOutput** component to the first **tAccessInput** component using an **OnComponentOk** link.

• In each of the **tAccessInput** components’ **Basic settings** view, select the **Use an existing connection** check box to retrieve the distributed data flow. Then set the schema manually through **[Edit schema]** dialog box.

• Then set the **Table Name** accordingly. In **tAccessInput_1**, this will be *Name*.

• Click on the **Guess Query**.

• Connect each **tAccessInput** component to **tLogRow** component with a **Row > Main** link. In each of the **tLogRow** components’ **basic settings** view, select **Table** in the **Mode** field.

• Add the **tAccessCommit** component below the **tFileList** component in the design workspace and connect them together using an **OnComponentOk** link in order to terminate the Job with the transaction commit.

• In the **basic settings** view of **tAccessCommit** component and from the **Component list**, select the connection to be used, **tAccessConnection_1** in this scenario.

• Save your Job and press **F6** to execute it.
The parent table Table1 is reused to generate the Name table and Birthday table.
**tAccessInput**

Reads a database and extracts fields based on a query.

tAccessInput executes a DB query with a strictly defined statement which must correspond to the schema definition. Then it passes on the field list to the next component via a Row > Main connection.

**tAccessInput Standard properties**

These properties are used to configure tAccessInput running in the Standard Job framework.

The Standard tAccessInput component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.

For more information about setting up and storing database connection parameters, see the section describing how to set up a DB connection of Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
</tbody>
</table>
1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>DB Version</strong></th>
<th>Select the version of Access that you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: The schema is created and stored locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>- <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td><strong>Query type and Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
</tbody>
</table>
## Advanced settings

<table>
<thead>
<tr>
<th><strong>Additional JDBC parameters</strong></th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <strong>Use an existing connection</strong> check box in the <strong>Basic</strong> settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
<tr>
<td><strong>Trim all the String/Char columns</strong></td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</td>
</tr>
<tr>
<td><strong>Trim column</strong></td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
</tbody>
</table>

## Global Variables

| **Global Variables** | **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see **Talend Studio User Guide**. |

## Usage

| **Usage rule** | This component offers the flexibility benefit of the DB query and covers all possible SQL queries. |
| **Dynamic settings** | Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  
The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defin |
ed, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Limitation**

<table>
<thead>
<tr>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you are using an ODBC driver, you need to work with Java 7, and make sure that your JVM and ODBC versions match up: both 64-bit or 32-bit. When working with Java 8, this component supports only the General collation mode of Access.</td>
</tr>
</tbody>
</table>

**Related scenarios**

For related topics, see:

Related topic in description of **tContextLoad** on page 487.
tAccessOutput

Writes, updates, makes changes or suppresses entries in a database.
tAccessOutput executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tAccessOutput Standard properties

These properties are used to configure tAccessOutput running in the Standard Job framework.
The Standard tAccessOutput component belongs to the Databases family.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.</td>
<td></td>
</tr>
<tr>
<td>For more information about setting up and storing database connection parameters, see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>

Use an existing connection

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component</td>
<td></td>
</tr>
</tbody>
</table>
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>DB Version</th>
<th>Select the version of Access that you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td><strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Create table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td><strong>Create table if not exists</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop table if exists and create</strong>: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td>Action on data</td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, Job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Make changes to existing entries.</td>
</tr>
<tr>
<td></td>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td><strong>Update or insert</strong>: Update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Remove entries corresponding to the input flow.</td>
</tr>
</tbody>
</table>

**Warning:**

You must specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking *Edit Schema* and selecting the check box(es) next to the column(s) you want to set as primary.
key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the update and delete operations. To do that: Select the **Use field options** check box and then in the **Key in update** column, select the check boxes next to the column name on which you want to base the update operation. Do the same in the **Key in delete** column for the deletion operation.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

**Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](https://help.talend.com).

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](https://help.talend.com).

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Die on error

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

### Advanced settings

| Additional JDBC parameters | Specify additional connection properties for the DB connection you are creating. This option is not available |
if you have selected the **Use an existing connection** check box in the **Basic** settings.

**Note:**
You can press **Ctrl+Space** to access a list of predefined global variables.

### Commit every

Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at executions.

### Additional Columns

This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.

- **Name:** Type in the name of the schema column to be altered or inserted as new column
- **SQL expression:** Type in the SQL statement to be executed in order to alter or insert the relevant column data.
- **Position:** Select **Before**, **Replace** or **After** following the action to be performed on the reference column.
- **Reference column:** Type in a column of reference that the **tDBOutput** can use to place or replace the new or altered column.

### tStat Catcher Statistics

Select this check box to collect log data at the component level.

### Use field options

Select this check box to customize a request, especially when there is double action on data.

### Enable debug mode

Select this check box to display each step during processing entries in a database.

### Support null in “SQL WHERE” statement

Select this check box if you want to deal with the Null values contained in a DB table.

**Note:**
Make sure the Nullable check box is selected for the corresponding columns in the schema.

### Global Variables

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>NB_LINE:</strong> the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_UPDATED:</strong> the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED:</strong> the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>
### Usage

#### Usage rule

This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible. This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in an Access database. It also allows you to create a reject flow using a Row > Rejects link to filter data in error. For an example of tMysqlOutput in use, see Scenario: Retrieving data in error with a Reject link on page 2529.

#### Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.
Limitation

If you are using an ODBC driver, you need to work with Java 7, and make sure that your JVM and ODBC versions match up: both 64-bit or 32-bit.

When working with Java 8, this component supports only the General collation mode of Access.

**Related scenarios**

For related topics, see:

- **Scenario: Inserting a column and altering data using tMysqlOutput** on page 2521.
tAccessOutputBulk

Prepares the file which contains the data used to feed the Access database.

The tAccessOutputBulk and tAccessBulkExec components are generally used together to output data to a delimited file and then to perform various actions on the file in an Access database, in a two step process. These two steps are fused together in the tAccessOutputBulkExec component, detailed in a separate section. The advantage of using a two step process is that it makes it possible to carry out transformations on the data before loading it in the database.

tAccessOutputBulk writes a delimited file.

**tAccessOutputBulk Standard properties**

These properties are used to configure tAccessOutputBulk running in the Standard Job framework.

The Standard tAccessOutputBulk component belongs to the Databases family.

The component in this framework is generally available.

⚠️ **Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>File Name</strong></td>
<td>Name and path to the file to be created and/or the variable to be used. For further information about how to define and use a variable in a Job, see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
<tr>
<td><strong>Create directory if not exists</strong></td>
<td>Select this check box to create the as yet non-existant file directory that specified in the <strong>File name</strong> field.</td>
</tr>
<tr>
<td><strong>Append</strong></td>
<td>Select this check box to add any new rows to the end of the file.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
</tbody>
</table>
**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-In** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

### Advanced settings

<table>
<thead>
<tr>
<th>Include header</th>
<th>Select this check box to include the column header in the file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding</td>
<td>Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

- **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with tAccessBulkExe component. Used together they offer gains in performance while feeding an Access database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component family</td>
<td>Databases/Access</td>
</tr>
<tr>
<td>Limitation</td>
<td>If you are using an ODBC driver, you need to work with Java 7, and make sure that your JVM and ODBC versions match up: both 64-bit or 32-bit. When working with Java 8, this component supports only the General collation mode of Access.</td>
</tr>
</tbody>
</table>

### Related scenarios

For use cases in relation with tAccessOutputBulk, see the following scenarios:

- Scenario: Inserting transformed data in MySQL database on page 2537
- Scenario: Inserting data in bulk in MySQL database on page 2544
tAccessOutputBulkExec

Executes an Insert action on the data provided, in an Access database.

The tAccessOutputBulk and tAccessBulkExec components are generally used together to output data to a delimited file and then to perform various actions on the file in an Access database, in a two step process. These two steps are fused together in tAccessOutputBulkExec.

As a dedicated component, tAccessOutputBulkExec improves performance during Insert operations in an Access database.

tAccessOutputBulkExec Standard properties

These properties are used to configure tAccessOutputBulkExec running in the Standard Job framework.

The Standard tAccessOutputBulkExec component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>Note: When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
<td></td>
</tr>
<tr>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
<td></td>
</tr>
</tbody>
</table>
2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

| **DB Version** | Select the version of Access that you are using. |
| **DB name** | Name of the database |
| **Username** and **Password** | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Action on table** | On the table defined, you can perform one of the following operations:  
- **None**: No operation is carried out.  
- **Drop and create table**: The table is removed and created again.  
- **Create table**: The table does not exist and gets created.  
- **Create table if not exists**: The table is created if it does not already exist.  
- **Drop table if exists and create**: The table is removed if it already exists and created again.  
- **Clear table**: The table content is deleted. |
| **Table** | Name of the table to be written.  
**Note:** Note that only one table can be written at a time and that the table must already exist for the insert operation to succeed |
| **FileName** | Name of the file to be processed. Related topic: see *Talend Studio User Guide*. |
| **Action on data** | On the data of the table defined, you can perform:  
- **Insert**: Add new entries to the table. |
| **Schema** and **Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.  
**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various
projects and Job designs. Related topic: see Talend Studio User Guide.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

### Advanced settings

| **Create directory if not exists** | Select this check box to create the as yet non existant file directory specified in the File name field. |
| **Append** | Select this check box to append new rows to the end of the file. |

#### Additional JDBC parameters

Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the Use an existing connection check box in the Basic settings.

**Note:**

You can press Ctrl+Space to access a list of predefined global variables.

| **Include header** | Select this check box to include the column header to the file. |
| **Encoding** | Select the encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling. |
| **tStatCatcher Statistics** | Select this check box to collect the log data at the component level. |
**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is mainly used when no particular transformation is required on the data to be loaded in the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Limitation</td>
<td>If you are using an ODBC driver, make sure that your JVM and ODBC versions match up: both 64-bit or 32-bit.</td>
</tr>
</tbody>
</table>

**Related scenarios**

For use cases in relation with tAccessOutputBulkExec, see the following scenarios:

- Scenario: Inserting data in bulk in MySQL database on page 2544
- Scenario: Inserting transformed data in MySQL database on page 2537
tAccessRollback

Cancels the transaction commit in the connected database and avoids to commit part of a transaction involuntarily.

tAccessRollback Standard properties

These properties are used to configure tAccessRollback running in the Standard Job framework.
The Standard tAccessRollback component belongs to the Databases family.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tAccessConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStat Catcher Statistics  | Select this check box to collect log data at the component level. |

Usage

| Usage rule | This component is more commonly used with other tAccess* components, especially with the tAccessConnection and tAccessCommit components. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |
Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

| Limitation | If you are using an ODBC driver, you need to work with Java 7, and make sure that your JVM and ODBC versions match up: both 64-bit or 32-bit. |

### Related scenarios

No scenario is available for the Standard version of this component yet.
tAccessRow

Executes the SQL query stated onto the specified database.

Depending on the nature of the query and the database, tAccessRow acts on the actual DB structure or on the data (although without handling data). The SQLBuilder tool helps you write easily your SQL statements. tAccessRow is the specific component for this database query. The row suffix means the component implements a flow in the job design although it does not provide output.

**tAccessRow Standard properties**

These properties are used to configure tAccessRow running in the Standard Job framework.

The Standard tAccessRow component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

| **DB Version** | Select the Access database version that you are using. |
| **Database** | Name of the database |
| **Username and Password** | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. |
| **Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*. |  |
| **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*. |  |
| **Table Name** | Name of the source table where changes made to data should be captured. |
| **Query type** | The query can be **Built-in** for a particular Job, or for commonly used query, it can be stored in the **Repository** to ease the query reuse. |
| **Built-in**: Fill in manually the query statement or build it graphically using SQLBuilder |  |
| **Repository**: Select the relevant query stored in the Repository. The Query field gets accordingly filled in. |  |
**Query**

Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

**Commit every**

Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.

**Die on error**

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row > Rejects link.

### Advanced settings

**Propagate QUERY’s recordset**

Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the use column list.

**Use PreparedStatement**

Select this check box if you want to query the database using a PreparedStatement. In the Set PreparedStatement Parameter table, define the parameters represented by "?" in the SQL instruction of the Query field in the Basic Settings tab.

- **Parameter Index**: Enter the parameter position in the SQL instruction.
- **Parameter Type**: Enter the parameter type.
- **Parameter Value**: Enter the parameter value.

*Note:*

This option is very useful if you need to execute the same query several times. Performance levels are increased.

**Commit every**

Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.

**tStat Catcher Statistics**

Select this check box to collect log data at the component level.

### Global Variables

**Global Variables**

- **QUERY**: the query statement being processed. This is a Flow variable and it returns a string.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the <strong>Use an existing connection</strong> check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Limitation

If you are using an ODBC driver, you need to work with Java 7, and make sure that your JVM and ODBC versions match up: both 64-bit or 32-bit. When working with Java 8, this component supports only the General collation mode of Access.

## Related scenarios

For related topics, see:

- Procedure on page 625
- Scenario: Removing and regenerating a MySQL table index on page 2552.
tAddCRCRow

Provides a unique ID which helps improving the quality of processed data.

tAddCRCRow calculates a surrogate key based on one or several columns and adds it to the defined schema.

**tAddCRCRow Standard properties**

These properties are used to configure tAddCRCRow running in the Standard Job framework.
The Standard tAddCRCRow component belongs to the Data Quality family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description. It defines the number of fields to be processed and passed on to the next component. The schema is either <strong>Built-in</strong> or stored remotely in the <strong>Repository</strong>. In this component, a new CRC column is automatically added.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>The schema will be created and stored locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>The schema already exists and is stored in the Repository, hence can be reused in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td>Implication</td>
<td>Select the check box facing the relevant columns to be used for the surrogate key checksum.</td>
</tr>
</tbody>
</table>

**Advanced Settings**

<table>
<thead>
<tr>
<th>CRC type</th>
<th>Select a CRC type in the list. The longer the CRC, the least overlap you will have.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions |
|------------------|-----------------------------------------------------------------------------------|

100
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is an intermediary step. It requires an input flow as well as an output.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the <strong>Install</strong> button on the <strong>Component</strong> tab view. You can also find out and add all missing JARs easily on the <strong>Modules</strong> tab in the <strong>Integration</strong> perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

### Scenario: Adding a surrogate key to a file

This scenario describes a Job adding a surrogate key to a delimited file schema.

![Diagram](image)

**Setting up the Job**

**Procedure**

1. Drop the following components: **tFileInputDelimited**, **tAddCRCRow** and **tLogRow**.
2. Connect them using a **Main row** connection.

**Configuring the input component**

**Procedure**

1. In the **tFileInputDelimited Component** view, set the **File Name** path and all related properties in case these are not stored in the **Repository**.
2. Create the schema through the Edit Schema button, if the schema is not stored already in the Repository. Remember to set the data type column and for more information on the Date pattern to be filled in, visit http://docs.oracle.com/javase/6/docs/api/index.html.

**Configuring the tAddCRCRow component**

**Procedure**

1. In the tAddCRCRow Component view, select the check boxes of the input flow columns to be used to calculate the CRC.

   Notice that a CRC column (read-only) has been added at the end of the schema.

2. Select **CRC32** as CRC Type to get a longer surrogate key.

3. In the Basic settings view of tLogRow, select the Print values in cells of a table option to display the output data in a table on the Console.

**Job execution**

Then save your Job and press F6 to execute it.
An additional CRC Column has been added to the schema calculated on all previously selected columns (in this case all columns of the schema).
tAddLocationFromIP

Replaces IP addresses with geographical locations.

tAddLocationFromIP geolocates visitors through their IP addresses: this component identifies visitors’ geographical locations (country, region, city, latitude, longitude, ZIP code, etc.) using an IP address lookup database file.

**tAddLocationFromIP Standard properties**

These properties are used to configure tAddLocationFromIP running in the Standard Job framework.
The Standard tAddLocationFromIP component belongs to the Misc family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description, it defines the fields to be processed and passed on to the next component. The schema of this component is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in:</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Repository:</strong></td>
<td>Select the Repository file where Properties are stored. When selected, the fields that follow are predefined using fetched data.</td>
</tr>
</tbody>
</table>

**Database Filepath**
The path to the IP address lookup database file.

**Input parameters**

<table>
<thead>
<tr>
<th><strong>Input column:</strong></th>
<th>Select the input column from which the input values are to be taken.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>input value is a hostname:</strong></td>
<td>Check if the input column holds hostnames.</td>
</tr>
<tr>
<td><strong>input value is an IP address:</strong></td>
<td>Check if the input column holds IP addresses.</td>
</tr>
</tbody>
</table>

**Location type**

<table>
<thead>
<tr>
<th><strong>Country code:</strong></th>
<th>Check to replace IP with country code.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Country name:</strong></td>
<td>Check to replace IP with country name.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |
Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is an intermediary step in the data flow allowing to replace IP with geolocation information. It can not be a start component as it requires an input flow. It also requires an output component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

Scenario: Identifying a real-world geographic location of an IP

The following scenario creates a three-component Job that associates an IP with a geographical location. It obtains a site visitor’s geographical location based on its IP.

Dropping and linking components

Procedure

1. Drop the following components from the Palette onto the design workspace: tFixedFlowInput, tAddLocationFromIP, and tLogRow.
2. Connect the three components using Row Main links.
Configuring the components

Procedure

1. In the design workspace, select tFixedFlowInput, and click the Component tab to define the basic settings for tFixedFlowInput.

2. Click the [...] button next to Edit Schema to define the structure of the data you want to use as input. In this scenario, the schema is made of one column that holds an IP address.

3. Click OK to close the dialog box, and accept propagating the changes when prompted by the system. The defined column is displayed in the Values panel of the Basic settings view.

4. In the Number of rows field, enter the number of rows to be generated, and click in the Value cell and set the value for the IP address.

5. In the design workspace, select tAddLocationFromIP and click the Component tab to define the basic settings for tAddLocationFromIP.
6. Click the **Sync columns** button to synchronize the schema with the input schema set with **tFixedFlowInput**.

7. Browse to the **GeoIP.dat** file to set its path in the **Database filepath** field.

   **Note:**
   
   Ensure to download the latest version of the IP address lookup database file from the relevant site as indicated in the **Basic settings** view of **tAddLocationFromIp**.

8. In the **Input parameters** panel, set your input parameters as needed. In this scenario, the input column is the **ip** column defined earlier that holds an IP address.

9. In the **Location type** panel, set location type as needed. In this scenario, we want to display the country name.

10. In the design workspace, select **tLogRow** and click the **Component** tab and define the basic settings for **tLogRow** as needed. In this scenario, we want to display values in cells of a table.

### Saving and executing the Job

#### Procedure

1. Press **Ctrl+S** to save your Job.

2. Press **F6** or click **Run** in the **Run** tab to execute the Job.

```
Starting job add_location at 11:00 06/08/2008.
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>tLogRow_1</td>
</tr>
<tr>
<td>ip</td>
</tr>
<tr>
<td>location</td>
</tr>
<tr>
<td>88.131.8.250</td>
</tr>
</tbody>
</table>
```

**Job add_location ended at 11:00 06/08/2008. [exit code=0]**

### Results

One row is generated to display the country name that is associated with the set IP address.
tAdvancedFileOutputXML

Writes an XML file with separated data values according to an XML tree structure.
tAdvancedFileOutputXML outputs data to an XML type of file and offers an interface to deal with loop and group by elements if needed.

tAdvancedFileOutputXML Standard properties

These properties are used to configure tAdvancedFileOutputXML running in the Standard Job framework.
The Standard tAdvancedFileOutputXML component belongs to the File and the XML families.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the Repository file where Properties are stored. The following fields are pre-filled in using fetched data.</td>
</tr>
</tbody>
</table>

Use Output Stream

Select this check box process the data flow of interest. Once you have selected it, the Output Stream field displays and you can type in the data flow of interest.

The data flow to be processed must be added to the flow in order for this component to fetch these data via the corresponding representative variable.

This variable could be already pre-defined in your Studio or provided by the context or the components you are using along with this component; otherwise, you could define it manually and use it according to the design of your Job, for example, using tJava or tJavaFlex.

In order to avoid the inconvenience of hand writing, you could select the variable of interest from the auto-completion list (Ctrl+Space) to fill the current field on condition that this variable has been properly defined.

For further information about how to use a stream, see Scenario 2: Reading data from a remote file in streaming mode on page 1015.

File name

Name or path to the output file and/or the variable to be used.

This field becomes unavailable once you have selected the Use Output Stream check box.

For further information about how to define and use a variable in a Job, see Talend Studio User Guide.
**Configure XML tree**
Opens the dedicated interface to help you set the XML mapping. For details about the interface, see *Defining the XML tree* on page 111.

**Schema and Edit Schema**
A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [**Repository Content**] window.

<table>
<thead>
<tr>
<th>Built-in</th>
<th>The schema will be created and stored locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>The schema already exists and is stored in the Repository, hence can be reused in various projects and job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

**Sync columns**
Click to synchronize the output file schema with the input file schema. The Sync function only displays once the Row connection is linked with the Output component.

**Append the source xml file**
Select this check box to add the new lines at the end of your source XML file.

**Generate compact file**
Select this check box to generate a file that does not have any empty space or line separators. All elements then are presented in a unique line and this will reduce considerably file size.

**Include DTD or XSL**
Select this check box to add the DOCTYPE declaration, indicating the root element, the access path and the DTD file, or to add the processing instruction, indicating the type of stylesheet used (such as XSL types), along with the access path and file name.

**Advanced settings**

<p>| Split output in several files | If the XML file output is big, you can split the file every certain number of rows. |</p>
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trim data</td>
<td>This check box is activated when you are using the dom4j generation mode. Select this check box to trim the leading or trailing whitespace from the value of a XML element.</td>
</tr>
<tr>
<td>Create directory only if not exists</td>
<td>This check box is selected by default. It creates a directory to hold the output XML files if required.</td>
</tr>
<tr>
<td>Create empty element if needed</td>
<td>This box is selected by default. If no column is associated to an XML node, this option will create an open/close tag in place of the expected tag.</td>
</tr>
<tr>
<td>Create attribute even if its value is NULL</td>
<td>Select this check box to generate XML tag attribute for the associated input column whose value is null.</td>
</tr>
<tr>
<td>Create attribute even if it is unmapped</td>
<td>Select this check box to generate XML tag attribute for the associated input column that is unmapped.</td>
</tr>
<tr>
<td>Create associated XSD file</td>
<td>If one of the XML elements is defined as a Namespace element, this option will create the corresponding XSD file.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> To use this option, you must select Dom4j as the generation mode.</td>
</tr>
<tr>
<td>Add Document type as node</td>
<td>Select this check box to add column(s) of the Document type as node(s) instead of escaped string(s) in the output XML file.</td>
</tr>
<tr>
<td></td>
<td>This check box appears only when the generation mode is set to <strong>Slow and memory-consuming (Dom4j)</strong> in the Advanced settings tab.</td>
</tr>
<tr>
<td>Advanced separator (for number)</td>
<td>Select this check box to change the expected data separator.</td>
</tr>
<tr>
<td></td>
<td><strong>Thousands separator:</strong> define the thousands separator, between inverted commas</td>
</tr>
<tr>
<td></td>
<td><strong>Decimal separator:</strong> define the decimals separator between inverted commas</td>
</tr>
<tr>
<td>Generation mode</td>
<td>Select the appropriate generation mode according to your memory availability. The available modes are:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Slow and memory-consuming (Dom4j)</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This option allows you to use dom4j to process the XML files of high complexity.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Fast with low memory consumption</strong></td>
</tr>
<tr>
<td></td>
<td>Once you select <strong>Append the source xml file</strong> in the Basic settings view, this field disappears because in this situation, your generation mode is set automatically as dom4j.</td>
</tr>
</tbody>
</table>
### Global Variables

**Global Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td><strong>NB_LINE</strong></td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

### Usage

**Usage rule**

Use this component to write an XML file with data passed on from other components using a Row link.

### Defining the XML tree

Double-click on the **tAdvancedFileOutputXML** component to open the dedicated interface or click on the three-dot button on the **Basic settings** vertical tab of the **Component Settings** tab.
To the left of the mapping interface, under **Schema List**, all of the columns retrieved from the incoming data flow are listed (only if an input flow is connected to the **tAdvancedFileOutputXML** component).

To the right of the interface, define the XML structure you want to obtain as output.

You can easily import the XML structure or create it manually, then map the input schema columns onto each corresponding element of the XML tree.

### Importing the XML tree

The easiest and most common way to fill out the XML tree panel, is to import a well-formed XML file.

**Procedure**

1. Rename the **root tag** that displays by default on the **XML tree** panel, by clicking on it once.
2. Right-click on the root tag to display the contextual menu.
3. On the menu, select **Import XML tree**.
4. Browse to the file to import and click **OK**.
   - You can import an XML tree from files in XML, XSD and DTD formats.
   - When importing an XML tree structure from an XSD file, you can choose an element as the root of your XML tree.

   The **XML Tree** column is hence automatically filled out with the correct elements.
5. If you need to add or remove an element or sub-elements, right-click the relevant element of the tree to display the contextual menu.
6. Select **Delete** to remove the selection from the tree or select the relevant option among: **Add sub-element**, **Add attribute**, **Add namespace** to enrich the tree.
Creating the XML tree manually

If you don’t have any XML structure defined as yet, you can create it manually.

Procedure
1. Rename the root tag that displays by default on the XML tree panel, by clicking on it once.
2. Right-click on the root tag to display the contextual menu.
3. On the menu, select Add sub-element to create the first element of the structure.
4. If you need to add an attribute or a child element to any element or remove any element, right-click the left of the corresponding element name to display the contextual menu.
5. Right-click to the left of the element name to display the contextual menu.
6. On the menu, select the relevant option among: Add sub-element, Add attribute, Add namespace or Delete.

Mapping XML data

Once your XML tree is ready, you can map each input column with the relevant XML tree element or sub-element to fill out the Related Column.

Procedure
1. Click on one of the Schema column name.
2. Drag it onto the relevant sub-element to the right.
3. Release to implement the actual mapping.
4. If you need to disconnect any mapping on any element of the XML tree, select the element and right-click to the left of the element name to display the contextual menu.
5. Select Disconnect linker.

Defining the node status

Defining the XML tree and mapping the data is not sufficient. You also need to define the loop element and if required the group element.

Define a loop element

The loop element allows you to define the iterating object. Generally the Loop element is also the row generator.

About this task
To define an element as loop element:

Procedure
1. Select the relevant element on the XML tree.
2. Right-click to the left of the element name to display the contextual menu.
3. Select Set as Loop Element.
The **Node Status** column shows the newly added status.

There can only be one loop element at a time.

**Define a group element**

The group element is optional, it represents a constant element where the groupby operation can be performed. A group element can be defined only if a loop element was defined before.

**About this task**

When using a group element, the rows should sorted, in order to be able to group by the selected node.

To define an element as group element:

**Procedure**

1. Select the relevant element on the XML tree.
2. Right-click to the left of the element name to display the contextual menu.
3. Select **Set as Group Element**.

**Results**

The **Node Status** column shows the newly added status and any group status required are automatically defined, if needed.

Click **OK** once the mapping is complete to validate the definition and continue the job configuration where needed.

**Scenario: Creating an XML file using a loop**

The following scenario describes the creation of an XML file from a sorted flat file gathering a video collection.

**Configuring the source file**

**Procedure**

1. Drop a **tFileInputDelimited** and a **tAdvancedFileOutputXML** from the Palette onto the design workspace.
2. Alternatively, if you configured a description for the input delimited file in the **Metadata** area of the **Repository**, then you can directly drag & drop the metadata entry onto the editor, to set up automatically the input flow.
3. Right-click on the input component and drag a row main link towards the **tAdvancedFileOutputXML** component to implement a connection.
4. Select the **tFileInputDelimited** component and display the **Component settings** tab located in the tab system at the bottom of the Studio.

![Component settings tab](image)

5. Select the **Property type**, according to whether you stored the file description in the Repository or not. If you dragged & dropped the component directly from the Metadata, no changes to the settings should be needed.

If you didn’t setup the file description in the **Repository**, then select **Built-in** and manually fill out the fields displayed on the **Basic settings** vertical tab.

The input file contains the following type of columns separated by semi-colons: `id, name, category, year, language, director and cast`.

```
<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>id;name;category;year;language;director;cast</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1;Tootsie;Comedy;1982;English;Sydney Pollack;Dustin Hoffman</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1;Tootsie;Comedy;1982;English;Sydney Pollack;Jessica Lange</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1;Tootsie;Comedy;1982;English;Sydney Pollack;Sydney Pollack</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2;The 5th Element;Science Fiction;1997;French;Eric Besson;Bruce Willis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2;The 5th Element;Science Fiction;1997;French;Eric Besson;Gary Oldman</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2;The 5th Element;Science Fiction;1997;French;Eric Besson;Milla Jovovich</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3;Leon, the Professional;Action drama;1994;French;Eric Besson;Jean Reno</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

In this simple use case, the **Cast** field gathers different values and the id increments when changing movie.

6. If needed, define the **tFileDelimitedInput** schema according to the file structure.
7. Once you checked that the schema of the input file meets your expectation, click on OK to validate.

**Configuring the XML output and mapping**

**Procedure**

1. Then select the tAdvancedFileOutputXML component and click on the Component settings tab to configure the basic settings as well as the mapping. Note that a double-click on the component will open directly the mapping interface.

2. In the File Name field, browse to the file to be written if it exists or type in the path and file name that needs to be created for the output.

   By default, the schema (file description) is automatically propagated from the input flow. But you can edit it if you need.

3. Then click on the three-dot button or double-click on the tAdvancedFileOutputXML component on the design workspace to open the dedicated mapping editor.

   To the left of the interface, are listed the columns from the input file description.

4. To the right of the interface, set the XML tree panel to reflect the expected XML structure output.

   You can create the structure node by node. For more information about the manual creation of an XML tree, see Defining the XML tree on page 111.

   In this example, an XML template is used to populate the XML tree automatically.
5. Right-click on the root tag displaying by default and select **Import XML tree** at the end of the contextual menu options.

6. Browse to the XML file to be imported and click **OK** to validate the import operation.

   **Note:**
   You can import an XML tree from files in XML, XSD and DTD formats.

7. Then drag & drop each column name from the **Schema List** to the matching (or relevant) **XML tree** elements as described in **Mapping XML data** on page 113.

   The mapping is shown as blue links between the left and right panels.

   ![XML tree mapping](image)

   Finally, define the node status where the loop should take place. In this use case, the **Cast** being the changing element on which the iteration should operate, this element will be the loop element.

   Right-click on the Cast element on the XML tree, and select **Set as loop element**.

8. To group by movie, this use case needs also a group element to be defined.

   Right-click on the Movie parent node of the XML tree, and select **Set as group element**.

   The newly defined node status show on the corresponding element lines.

9. Click **OK** to validate the configuration.

10. Press **F6** to execute the Job.
The output XML file shows the structure as defined.
**tAggregateRow**

Receives a flow and aggregates it based on one or more columns.

For each output line, are provided the aggregation key and the relevant result of set operations (min, max, sum...).

`tAggregateRow` helps to provide a set of metrics based on values or calculations.

**tAggregateRow Standard properties**

These properties are used to configure `tAggregateRow` running in the Standard Job framework.

The Standard `tAggregateRow` component belongs to the Processing family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.  

| Built-In: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**. |  |
| Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**. |  |

| Group by | Define the aggregation sets, the values of which will be used for calculations. |
| Output Column: Select the column label in the list offered based on the schema structure you defined. You can add as many output columns as you wish to make more precise aggregations. |  |
Ex: Select Country to calculate an average of values for each country of a list or select Country and Region if you want to compare one country’s regions with another country’s regions.

**Input Column**: Match the input column label with your output columns, in case the output label of the aggregation set needs to be different.

**Operations**
Select the type of operation along with the value to use for the calculation and the output field.

**Output Column**: Select the destination field in the list.

**Function**: Select the operator among:
- **count**: calculates the number of rows
- **min**: selects the minimum value
- **max**: selects the maximum value
- **avg**: calculates the average
- **sum**: calculates the sum
- **first**: returns the first value
- **last**: returns the last value
- **list**: lists values of an aggregation by multiple keys.
- **list (object)**: lists Java values of an aggregation by multiple keys
- **count (distinct)**: counts the number of the distinct rows
- **standard deviation**: calculates the variability
- **union (geometry)**: makes the union of a set of Geometry objects

**Input column**: Select the input column from which the values are taken to be aggregated.

**Ignore null values**: Select the check boxes corresponding to the names of the columns for which you want the NULL value to be ignored.

### Advanced settings

**Delimiter (only for list operation)**
Enter the delimiter you want to use to separate the different operations.

**Use financial precision, this is the max precision for "sum" and "avg" operations, checked option heaps more memory and slower than unchecked.**
Select this check box to use a financial precision. This is a max precision but consumes more memory and slows the processing.

⚠️ **Warning:**
*We advise you to use the BigDecimal type for the output in order to obtain precise results.*
Check type overflow (slower)
Checks the type of data to ensure that the Job doesn’t crash.

Check ULP (Unit in the Last Place), ensure that a value will be incremented or decremented correctly, only float and double types. (slower)
Select this check box to ensure the most precise results possible for the Float and Double types.

tStatCatcher Statistics
Check this box to collect the log data at component level. Note that this check box is not available in the Map/Reduce version of the component.

Global Variables

Global Variables

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule
This component handles flow of data therefore it requires input and output, hence is defined as an intermediary step. Usually the use of tAggregateRow is combined with the tSortRow component.

Aggregating values and sorting data

This example shows you how to use Talend components to aggregate the students’ comprehensive scores and then sort the aggregated scores based on the student names.

Creating a Job for aggregating and sorting data

Create a Job to aggregate the students’ comprehensive scores using the tAggregateRow component, then sort the aggregated data using the tSortRow component, finally display the aggregated and sorted data on the console.
Procedure

1. Create a new Job and add a `tFixedFlowInput` component, a `tAggregateRow` component, a `tSortRow` component, and a `tLogRow` component by typing their names in the design workspace or dropping them from the Palette.

2. Link the `tFixedFlowInput` component to the `tAggregateRow` component using a Row > Main connection.

3. Do the same to link the `tAggregateRow` component to the `tSortRow` component, and the `tSortRow` component to the `tLogRow` component.

Configuring the Job for aggregating and sorting data

Configure the Job to aggregate the students’ comprehensive scores using the `tAggregateRow` component and then sort the aggregated data using the `tSortRow` component.

Procedure

1. Double-click the `tFixedFlowInput` component to open its Basic settings view.

2. Click the button next to Edit schema to open the schema dialog box and define the schema by adding two columns, `name` of String type and `score` of Double type. When done, click OK to save the changes and close the schema dialog box.

3. In the Mode area, select Use Inline Content (delimited file) and in the Content field displayed, enter the following input data:

   Peter;92
   James;93
   Thomas;91
   Peter;94
   James;96
   Thomas;95
   Peter;96
   James;92
   Thomas;98
   Peter;95
   James;96
   Thomas;93
   Peter;98
   James;97
   Thomas;95

4. Double-click the `tAggregateRow` component to open its Basic settings view.
5. Click the button next to Edit schema to open the schema dialog box and define the schema by adding five columns, name of String type, and sum, average, max, and min of Double type.

When done, click OK to save the changes and close the schema dialog box.

6. Add one row in the Group by table by clicking the button below it, and select name from both the Output column and Input column position column fields to group the input data by the name column.

7. Add four rows in the Operations table and define the operations to be carried out. In this example, the operations are sum, average, max, and min. Then select score from all four Input column position column fields to aggregate the input data based on it.

8. Double-click the tSortRow component to open its Basic settings view.
9. Add one row in the Criteria table and specify the column based on which the sort operation is performed. In this example, it is the name column. Then select alpha from the sort num or alpha? column field and asc from the Order asc or desc? column field to sort the aggregated data in ascending alphabetical order.

10. Double-click the tLogRow component to open its Basic settings view, and then select Table (print values in cells of a table) in the Mode area for better readability of the result.

**Executing the Job to aggregate and sort data**

After setting up the Job and configuring the components used in the Job for aggregating and sorting data, you can then execute the Job and verify the Job execution result.

**Procedure**

1. Press Ctrl + S to save the Job.
2. Press F6 to execute the Job.

```
[statistics] connecting to socket on port 3914
[statistics] connected
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>tLogRow_1</td>
</tr>
<tr>
<td>--------------------------</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>--------------------------</td>
</tr>
<tr>
<td>James</td>
</tr>
<tr>
<td>Peter</td>
</tr>
<tr>
<td>Thomas</td>
</tr>
</tbody>
</table>
[statistics] disconnected
```

**Results**

As shown above, the students' comprehensive scores are aggregated and then sorted in ascending alphabetical order based on the student names.
tAggregateSortedRow

Receives a sorted flow and aggregates it based on one or more columns.

For each output line, are provided the aggregation key and the relevant result of set operations (min, max, sum...). As the input flow is meant to be sorted already, the performance are hence greatly optimized.

tAggregateSortedRow helps to provide a set of metrics based on values or calculations.

tAggregateSortedRow Standard properties

These properties are used to configure tAggregateSortedRow running in the Standard Job framework.

The Standard tAggregateSortedRow component belongs to the Processing family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td>Click Sync columns to retrieve the schema from the previous component connected in the Job.</td>
<td></td>
</tr>
<tr>
<td>Built-in: The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Repository: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job flowcharts. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Input rows count</td>
<td>Specify the number of rows that are sent to the tAggregateSortedRow component.</td>
</tr>
</tbody>
</table>
### Group by
Define the aggregation sets, the values of which will be used for calculations.

**Output Column:** Select the column label in the list offered based on the schema structure you defined. You can add as many output columns as you wish to make more precise aggregations.

Ex: Select Country to calculate an average of values for each country of a list or select Country and Region if you want to compare one country’s regions with another country’s regions.

**Input Column:** Match the input column label with your output columns, in case the output label of the aggregation set needs to be different.

### Operations
Select the type of operation along with the value to use for the calculation and the output field.

**Output Column:** Select the destination field in the list.

**Function:** Select the operator among:

- **count:** calculates the number of rows
- **min:** selects the minimum value
- **max:** selects the maximum value
- **avg:** calculates the average
- **sum:** calculates the sum
- **first:** returns the first value
- **last:** returns the last value
- **list:** lists values of an aggregation by multiple keys.
- **list (object):** lists Java values of an aggregation by multiple keys
- **count (distinct):** counts the number of the distinct rows
- **standard deviation:** calculates the variability
- **union (geometry):** makes the union of a set of Geometry objects

**Input column:** Select the input column from which the values are taken to be aggregated.

**Ignore null values:** Select the check boxes corresponding to the names of the columns for which you want the NULL value to be ignored.
Advanced settings

| tStatCatcher Statistics | Check this box to collect the log data at component level. |

Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule | This component handles flow of data therefore it requires input and output, hence is defined as an intermediary step. |

Scenario: Sorting and aggregating the input data

This scenario describes a Job that sorts the entries of the input data based on two columns and displays the sorted data on the console, then aggregates the sorted data based on one column and displays the aggregated data on the console.

Adding and linking components

Procedure

1. Create a new Job and add the following components by typing their names in the design workspace or dropping them from the Palette: a tFixedFlowInput component, a tSortRow component, a tAggregateSortedRow component, and two tLogRow components.
2. Link tFixedFlowInput to tSortRow using a Row > Main connection.
3. Do the same to link tSortRow to the first tLogRow, link the first tLogRow to tAggregateSortedRow, and link tAggregateSortedRow to the second tLogRow.

Configuring the components

Sorting the input data

Procedure

1. Double-click tFixedFlowInput to open its Basic settings view.

2. Click the [...] button next to Edit schema and in the pop-up window define the schema by adding four columns: Id and Age of Integer type, and Name and Team of String type.

Click OK to close the schema editor and accept the propagation prompted by the pop-up dialog box.
3. In the Mode area, select **Use Inline Content(delimited file)**, and then in the Content field displayed, enter the input data to be sorted and aggregated. In this example, the input data is as follows:

```
1;Thomas;28;Component Team
2;Harry;32;Doc Team
3;John;26;Component Team
4;Nicolas;27;QA Team
5;George;24;Component Team
6;Peter;30;Doc Team
7;Teddy;23;QA Team
8;James;26;Component Team
```

4. Double-click **tSortRow** to open its Basic settings view.

5. Click the [+] button below the Criteria table to add as many rows as required and then specify the sorting criteria in the table. In this example, two rows are added, and the input entries will be sorted based on the column **Team** and then the column **Age**, both in ascending order.

6. Double-click the first **tLogRow** to open its Basic settings view.

7. In the Mode area, select **Table (print values in cells of a table)** for better readability of the sorting result.

**Aggregating the sorted data**

**Procedure**

1. Double-click **tAggregateSortedRow** to open its Basic settings view.
2. Click the [...] button next to Edit schema and in the pop-up window define the schema by adding five columns: AggTeam of String type, AggCount, MinAge, MaxAge, and AvgAge of Integer type.

Click OK to close the schema editor and accept the propagation prompted by the pop-up dialog box.

3. In the Input rows count field, enter the exact number of rows of the input data. In this example, it is 8.

4. Click the [+] button below the Group by table to add as many rows as required and specify the aggregation set in the table. In this example, the data will be aggregated based on the input column Team.

5. Click the [+] button below the Operations table to add as many rows as required and specify the operation to be carried out and the corresponding input column from which the data will be taken for each output column. In this example, we want to calculate the number of the input entries, the minimum age, the maximum age, and the average age for each team.

6. Double-click the second tLogRow to open its Basic settings view.
7. In the **Mode** area, select **Table (print values in cells of a table)** for better readability of the sorting result.

### Saving and executing the Job

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Execute the Job by pressing **F6** or clicking **Run** on the **Run** tab.

```
[statistics] connecting to socket on port 3701
[statistics] connected

<table>
<thead>
<tr>
<th>tLogRow_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

As shown above, the input entries are sorted based on the column *Team* and then the column *Age*, both in ascending order, and the sorted entries are then aggregated based on the column *Team*.
tAmazonAuroraClose

Closes an active connection to an Amazon Aurora database instance to release the occupied resources.

**tAmazonAuroraClose Standard properties**

These properties are used to configure tAmazonAuroraClose running in the Standard Job framework. The Standard tAmazonAuroraClose component belongs to the Cloud and the Databases families. The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the tAmazonAuroraConnection component that opens the connection you need to close from the list.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see [Talend Studio User Guide](#). |


Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other Amazon Aurora components, especially with the tAmazonAuroraConnection and tAmazonAuroraCommit components.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

Related scenario

For a related scenario, see Scenario: Handling data with Amazon Aurora on page 142.
tAmazonAuroraCommit

Commits in one go a global transaction instead of doing that on every row or every batch, and provides gain in performance, using a unique connection.

tAmazonAuroraCommit validates the data processed through the Job into the connected Amazon Aurora database.

**tAmazonAuroraCommit Standard properties**

These properties are used to configure tAmazonAuroraCommit running in the Standard Job framework.

The Standard tAmazonAuroraCommit component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the tAmazonAuroraConnection component for which you want the commit action to be performed.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default and it allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection after the component has performed its task.</td>
</tr>
</tbody>
</table>

**Warning:**

If you want to use a Row > Main connection to link tAmazonAuroraCommit to your Job, your data will be committed row by row. In this case, do not select the Close Connection check box or your connection will be closed before the end of your first row commit.

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |
### Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. 

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. 

For further information about variables, see *Talend Studio User Guide*. |

### Usage

| Usage rule | This component is more commonly used with other Amazon Aurora components, especially with the **tAmazonAuroraConnection** and **tAmazonAuroraRollback** components. |

| Dynamic settings | Click the [+ ] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. 

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable. 

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*. |

### Related scenario

For a related scenario, see *Scenario: Handling data with Amazon Aurora* on page 142.
tAmazonAuroraConnection

Opens a connection to an Amazon Aurora database instance that can then be reused by other Amazon Aurora components.

tAmazonAuroraConnection Standard properties

These properties are used to configure tAmazonAuroraConnection running in the Standard Job framework.

The Standard tAmazonAuroraConnection component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property Type</td>
<td>Either <strong>Built-In</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored.</td>
</tr>
<tr>
<td></td>
<td>The database connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Host</td>
<td>Type in the IP address or hostname of the Amazon Aurora database.</td>
</tr>
<tr>
<td>Port</td>
<td>Type in the listening port number of the Amazon Aurora database.</td>
</tr>
<tr>
<td>Database</td>
<td>Type in the name of the database you want to use.</td>
</tr>
<tr>
<td>Additional JDBC parameters</td>
<td>Specify additional connection properties for the database connection you are creating.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Type in the database user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td>Use or register a shared DB Connection</td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the <strong>Shared DB Connection Name</strong> field.</td>
</tr>
</tbody>
</table>
displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the `Use dynamic job` and `Use an independent process to run subjob` options of the `tRunJob` component. Using a shared connection together with a `tRunJob` component with either of these two options enabled will cause your Job to fail.

This check box is not available when the `Specify a data source alias` check box is selected.

Specify a data source alias

Select this check box and specify the alias of a data source created on the `Talend Runtime` side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in `Talend Runtime`. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

This check box disappears when the `Use or register a shared DB Connection` check box is selected.

Data source alias

Type in the alias of the data source created on the `Talend Runtime` side.

This field appears only when the `Specify a data source alias` check box is selected.

Advanced settings

Auto Commit

Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

`tStatCatcher Statistics`

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.
Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide.* |

Usage

| Usage rule | This component is more commonly used with other Amazon Aurora components, especially with the **tAmazonAuroraCommit** and **tAmazonAuroraRollback** components. |

Related scenario

For a related scenario, see *Scenario: Handling data with Amazon Aurora* on page 142.
tAmazonAuroraInput

Reads an Amazon Aurora database and extracts fields based on a query.

tAmazonAuroraInput executes a database query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Row ➔Main link.

**tAmazonAuroraInput Standard properties**

These properties are used to configure tAmazonAuroraInput running in the Standard Job framework.

The Standard tAmazonAuroraInput component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Either Built-In or Repository.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-In:</strong> No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
| **Host** | Type in the IP address or hostname of the Amazon Aurora database. |
| **Port** | Type in the listening port number of the Amazon Aurora database. |
| **Database** | Type in the name of the database you want to use. |
| **Username** and **Password** | Type in the database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Schema** and **Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. |
| **Table Name** | Type in the name of the table to be read. |
| **Query Type** and **Query** | Enter the database query paying particularly attention to the proper sequence of the fields in order to match the schema definition. |
| **Guess Query** | Click the button to generate the query which corresponds to the table schema in the Query field. |
Guess schema

Click the button to retrieve the schema from the table.

Specify a data source alias

Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

This check box disappears when the Use an existing connection check box is selected.

Data source alias

Type in the alias of the data source created on the Talend Runtime side.

This field appears only when the Specify a data source alias check box is selected.

Advanced settings

Additional JDBC parameters

Specify additional connection properties for the database connection you are creating. When you need to handle data of the time-stamp type 0000-00-00 00:00:00 using this component, set the parameter to noDateTimeStringSync=true&zeroDateTimeBehavior=convertToNull.

This field disappears when the Use an existing connection check box in the Basic settings view is selected.

Enable stream

Select this check box to enable streaming over buffering which allows the code to read from a large table without consuming a large amount of memory in order to optimize the performance.

Trim all the String/Char columns

Select this check box to remove leading and trailing whitespace from all the String/Char columns.

Trim column

Select the check box(es) in the Trim column to remove leading and trailing whitespace from the corresponding column(s).

This option disappears when the Trim all the String/Char columns check box is selected.

tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Global Variables

Global Variables

**NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.
**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is usually used as a start component of a Job or Subjob and it needs an output link.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamic settings</strong></td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The <strong>Dynamic settings</strong> table is available only when the <strong>Use an existing connection</strong> check box is selected in the <strong>Basic settings</strong> view. Once a dynamic parameter is defined, the <strong>Component List</strong> box in the <strong>Basic settings</strong> view becomes unusable. For examples on using dynamic parameters, see <strong>Scenario: Reading data from databases through context-based dynamic connections</strong> on page 2497 and <strong>Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters</strong> on page 489. For more information on <strong>Dynamic settings</strong> and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

**Scenario: Handling data with Amazon Aurora**

This scenario describes a Job that writes the user information into Amazon Aurora, and then reads the information in Amazon Aurora and displays it on the console.
The scenario requires the following seven components:

- **tAmazonAuroraConnection**: opens a connection to Amazon Aurora.
- **tFixedFlowInput**: defines the user information data structure, and sends the data to the next component.
- **tAmazonAuroraOutput**: writes the data it receives from the preceding component into Amazon Aurora.
- **tAmazonAuroraCommit**: commits in one go the data processed to Amazon Aurora.
- **tAmazonAuroraInput**: reads the data from Amazon Aurora.
- **tLogRow**: displays the data it receives from the preceding component on the console.
- **tAmazonAuroraClose**: closes the connection to Amazon Aurora.
**Adding and linking the components**

**Procedure**

1. Create a new Job and add seven components listed previously by typing their names in the design workspace or dropping them from the Palette.
2. Connect **tFixedFlowInput** to **tAmazonAuroraOutput** using a **Row > Main** connection.
3. Do the same to connect **tAmazonAuroraInput** to **tLogRow**.
4. Connect **tAmazonAuroraConnection** to **tFixedFlowInput** using a **Trigger > OnSubjobOk** connection.
5. Do the same to connect **tFixedFlowInput** to **tAmazonAuroraCommit**, **tAmazonAuroraCommit** to **tAmazonAuroraInput**, and **tAmazonAuroraInput** to **tAmazonAuroraClose**.

**Configuring the components**

**Opening a connection to Amazon Aurora**

**Procedure**

1. Double-click **tAmazonAuroraConnection** to open its **Basic settings** view.

![Amazon Aurora Connection Configuration](image)

2. In the **Host**, **Port**, **Database**, **Username** and **Password** fields, enter the information required for the connection to Amazon Aurora.

**Writing the data into Amazon Aurora**

**Procedure**

1. Double-click **tFixedFlowInput** to open its **Basic settings** view.
2. Click the [...] button next to Edit schema and in the pop-up window define the schema by adding three columns: `id` of Integer type, and `name` and `city` of String type.

![Schema of tFixedFlowInput_1](image)

Click OK to validate the changes and accept the propagation prompted by the pop-up dialog box.

3. In the Mode area, select Use Inline Content (delimited file) and enter the following user information in the Content field.

```
1;George;Bismarck
2;Abraham;Boise
3;Taylor;Nashville
4;William;Jefferson City
5;Kate;Jackson
6;James;Boise
7;Gerald;Little Rock
8;Chester;Richmond
9;Thomas;Springfield
10;Andre;Nashville
```

4. Double-click tAmazonAuroraOutput to open its Basic settings view.
5. Select the **Use an existing connection** check box and in the **Component List** that appears, select the connection component you have configured.

6. In the **Table** field, enter or browse to the table into which you want to write the data. In this example, it is `TalendUser`.

7. Select **Drop table if exists and create** from the **Action on table** drop-down list, and select **Insert** from the **Action on data** drop-down list.

8. Double-click **tAmazonAuroraCommit** to open its **Basic settings** view.

9. Clear the **Close Connection** check box if it is selected.

**Retrieving the data from Amazon Aurora**

**Procedure**

1. Double-click **tAmazonAuroraInput** to open its **Basic settings** view.

2. Select the **Use an existing connection** check box and in the **Component List** that appears, select the connection component you have configured.
3. Click the [...] button next to **Edit schema** and in the pop-up window define the schema by adding three columns: *id* of Integer type, and *name* and *city* of String type. The data structure is same as the structure you have defined for **tFixedFlowInput**.

4. In the **Table Name** field, enter or browse to the table into which you write the data. In this example, it is **TalendUser**.

5. Click the **Guess Query** button to generate the query. The **Query** field will be filled with the automatically generated query.

6. Double-click **tLogRow** to open its **Basic settings** view.

![tLogRow_1](image)

| Basic settings | | Edit schema | | Sync columns |
|---|---|---|---|
| Schema | Built-In | Edit schema | Sync columns |
| Advanced settings | | | |
| Dynamic settings | | | |
| View | | | |

7. In the **Mode** area, select **Table (print values in cells of a table)** for better readability of the result.

**Closing the connection to Amazon Aurora**

**Procedure**

1. Double-click **tAmazonAuroraClose** to open its **Basic settings** view.

![tAmazonAuroraClose_1](image)

<table>
<thead>
<tr>
<th>Basic settings</th>
<th>Component List</th>
<th>tAmazonAuroraConnection_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced settings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic settings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>View</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. In the **Component List**, select the connection component you have configured.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** or click **Run** on the **Run** tab to run the Job.
As shown above, the user information is written into Amazon Aurora, and then the data is retrieved from Amazon Aurora and displayed on the console.
tAmazonAuroraOutput

Writes, updates, makes changes or suppresses entries in an Amazon Aurora database.

tAmazonAuroraOutput executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

**tAmazonAuroraOutput Standard properties**

These properties are used to configure tAmazonAuroraOutput running in the Standard Job framework. The Standard tAmazonAuroraOutput component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Either <strong>Built-In</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-In:</strong> No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository:</strong> Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Type in the IP address or hostname of the Amazon Aurora database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Type in the listening port number of the Amazon Aurora database.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Type in the name of the database you want to use.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Type in the database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Type in the name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
</tbody>
</table>

**Action on table**

On the table defined, you can perform one of the following operations:

- **None**: No operation is carried out.
- **Drop and create table**: The table is removed and created again.
- **Create table**: The table does not exist and gets created.
- **Create table if not exists**: The table is created if it does not exist.
- **Drop table if exists and create**: The table is removed if it already exists and created again.
- **Clear table**: The table content is deleted.
- **Truncate table**: The table content is quickly deleted. However, you will not be able to rollback the operation.

**Action on data**

On the data of the table defined, you can perform one of the following operations:

- **Insert**: Add new entries to the table. If duplicates are found, the job stops.
- **Update**: Make changes to existing entries.
- **Insert or update**: Insert a new record. If the record with the given reference already exists, an update would be made.
- **Update or insert**: Update the record with the given reference. If the record does not exist, a new record would be inserted.
- **Delete**: Remove entries corresponding to the input flow.
- **Replace**: Add new entries to the table. If an old row in the table has the same value as a new row for a
**Amazon Aurora Output**

1. PRIMARY KEY or a UNIQUE index, the old row is deleted before the new row is inserted.

2. **Insert or update on duplicate key or unique index**: Add entries if the inserted value does not exist or update entries if the inserted value already exists and there is a risk of violating a unique index or primary key.

3. **Insert Ignore**: Add only new rows to prevent duplicate key errors.

**Warning:**

You must specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking **Edit schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the update and delete operations. To do that: Select the **Use field options** check box and then in the **Key in update** column, select the check boxes next to the column name on which you want to base the update operation. Do the same in the **Key in delete** column for the deletion operation.

---

### **Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and
decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Specify a data source alias</strong></td>
<td>Select this check box and specify the alias of a data source created on the <em>Talend Runtime</em> side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in <em>Talend Runtime</em>. For a related use case, see <a href="#">Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database</a> on page 2503. This check box disappears when the <strong>Use an existing connection</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Data source alias</strong></td>
<td>Type in the alias of the data source created on the <em>Talend Runtime</em> side. This field appears only when the <strong>Specify a data source alias</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row in error and complete the process for error-free rows. If needed, you can retrieve the rows in error via a <em>Row &gt; Rejects</em> link.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional JDBC parameters</strong></td>
<td>Specify additional connection properties for the database connection you are creating. This field disappears when the <strong>Use an existing connection</strong> check box in the <em>Basic settings</em> view is selected. <strong>Note:</strong> You can press <strong>Ctrl+Space</strong> to access a list of predefined global variables.</td>
</tr>
<tr>
<td><strong>Extend Insert</strong></td>
<td>Select this check box to carry out a bulk insert of a defined set of lines instead of inserting lines one by one. The gain in system performance is considerable. This check box appears only when the <strong>Insert</strong> option is selected from the <em>Action on data</em> list in the <em>Basic settings</em> view. <strong>Note:</strong> This option is not compatible with the <strong>Reject</strong> link. You should therefore clear the check box if you are using a <em>Row &gt; Rejects</em> link with this component.</td>
</tr>
<tr>
<td><strong>Number of rows per insert</strong></td>
<td>Enter the number of rows to be inserted per operation. Note that the higher the value specified, the lower performance levels shall be due to the increase in memory demands.</td>
</tr>
</tbody>
</table>
| **Use Batch** | Select this check box to activate the batch mode for data processing.  
This check box is available only when the **Update** or **Delete** option is selected from the **Action on data** list in the **Basic settings** view. |
| **Batch Size** | Specify the number of records to be processed in each batch.  
This field appears only when the **Use batch mode** check box is selected. |
| **Commit every** | Enter the number of rows to be included in a batch before it is committed to the database. This option ensures transaction quality (but not rollback) and, above all, a higher performance level. |
| **Additional columns** | This option allows you to call SQL functions to perform actions on columns, provided that these are not insert, update or delete actions, or actions that require preprocessing. This option is not available if you have just created the database table (even if you delete it beforehand). Click the [*] button under the table to add column(s), and set the following parameters for each column. |
| | • **Name**: Type in the name of the schema column to be altered or inserted. |
| | • **SQL expression**: Type in the SQL statement to be executed in order to alter or insert the data in the corresponding column. |
| | • **Position**: Select **Before**, **After** or **Replace** depending on the action to be performed on the reference column. |
| | • **Reference column**: Type in a reference column that **tAmazonAuroraOutput** can use to locate or replace the new column or the column to be modified. |
| **Use field options** | Select the check box for the corresponding column to customize a request, particularly if multiple actions are being carried out on the data. |
| | • **Key in update**: Select the check box for the corresponding column based on which the data is updated. |
| | • **Key in delete**: Select the check box for the corresponding column based on which the data is deleted. |
| | • **Updatable**: Select the check box if the data in the corresponding column can be updated. |
| | • **Insertable**: Select the check box if the data in the corresponding column can be inserted. |
### Use Hint Options
Select this check box to configure the hint(s) which can help you optimize a query’s execution.

<table>
<thead>
<tr>
<th>Hint Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click the [+] button under the table to add hint(s) and set the following parameters for each hint. This table appears only when the <strong>Use Hint Options</strong> check box is selected.</td>
</tr>
<tr>
<td>• <strong>HINT</strong>: Specify the hint you need, using the syntax <code>/* + */</code>.</td>
</tr>
<tr>
<td>• <strong>POSITION</strong>: Specify where you put the hint in an SQL statement.</td>
</tr>
<tr>
<td>• <strong>SQL STMT</strong>: Select an SQL statement <strong>INSERT</strong>, <strong>UPDATE</strong>, or <strong>DELETE</strong> you need to use.</td>
</tr>
</tbody>
</table>

### Enable debug mode
Select this check box to display each step involved in the process of writing data in the database.

### Use duplicate key update mode insert
Select this check box to activate the ON DUPLICATE KEY UPDATE mode, and then click the [+] button under the table displayed to add column(s) to be updated and specify the update action to be performed on the corresponding column.

| **Column** | Enter the name of the column to be updated. |
| **Value** | Enter the action to be performed on the column. |

This check box is available only when the **Insert** option is selected from the **Action on data** list in the **Basic settings** view.

### tStatCatcher Statistics
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_UPDATED</strong>: the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_DELETED</strong>: the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_REJECTED</strong>: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

#### Usage rule

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in an Amazon Aurora database. It also allows you to create a reject flow using a Row > Rejects link to filter data in error. For a similar scenario, see .

#### Dynamic settings

Click the [*] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Related scenario

For a related scenario, see *Scenario: Handling data with Amazon Aurora* on page 142.
tAmazonAuroraRollback

Rolls back any changes made in the Amazon Aurora database to prevent partial transaction commit if an error occurs.

**tAmazonAuroraRollback Standard properties**

These properties are used to configure tAmazonAuroraRollback running in the Standard Job framework.

The Standard tAmazonAuroraRollback component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

![Note]

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the <strong>tAmazonAuroraConnection</strong> component for which you want the rollback action to be performed.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default and it allows you to close the database connection once the rollback is done. Clear this check box to continue to use the selected connection after the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other Amazon Aurora components, especially with the <code>tAmazonAuroraConnection</code> and <code>tAmazonAuroraCommit</code> components.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.  

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*. |

### Related Scenario

No scenario is available for the Standard version of this component yet.
tAmazonEMRListInstances

Lists the details about the instance groups in a cluster on Amazon EMR (Elastic MapReduce).

**tAmazonEMRListInstances Standard properties**

These properties are used to configure tAmazonEMRListInstances running in the Standard Job framework.

The Standard tAmazonEMRListInstances component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

| **Access key** and **Secret key** | Specify the access keys (the access key ID in the **Access Key** field and the secret access key in the **Secret Key** field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key).

To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
|---|---|

| **Inherit credentials from AWS role** | Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances. |
| **Assume role** | Select this check box and specify the values for the following parameters used to create a new assumed role session.

- **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.

- **Role session name**: an identifier for the assumed role session.

- **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.

For more information on assuming roles, see AssumeRole. |
| **Region** | Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (for example 'us-east-1'). For more information about how to specify the AWS region, see Choose an AWS Region. |
**Filter master and core instances**
Select this check box to ignore the master and core instance groups and list only the task instance groups.

**Cluster id**
Enter the ID of the cluster for which you want to list the instance groups.

**Advanced settings**

**STS Endpoint**
Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.
This check box is available only when the Assume role check box is selected.

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

**Global Variables**

**CURRENT_GROUP_ID**: the ID of the current instance group. This is an After variable and it returns a string.

**CURRENT_GROUP_NAME**: the name of the current instance group. This is an After variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

**Usage rule**
tAmazonEMRListInstances is usually used as a start component of a Job or Subjob.

**Related scenario**

No scenario is available for the Standard version of this component yet.
tAmazonEMRManage

Launches or terminates a cluster on Amazon EMR (Elastic MapReduce).

**tAmazonEMRManage Standard properties**

These properties are used to configure tAmazonEMRManage running in the Standard Job framework.
The Standard tAmazonEMRManage component belongs to the Cloud family.
The component in this framework is generally available.

**Basic settings**

| Access key and Secret key | Specify the access keys (the access key ID in the Access Key field and the secret access key in the Secret Key field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key).
To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Inherit credentials from AWS role | Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances. |
| Assume role | Select this check box and specify the values for the following parameters used to create a new assumed role session.
• **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.
• **Role session name**: an identifier for the assumed role session.
• **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.
For more information on assuming roles, see AssumeRole. |
| Action | Select an action to be performed from the list, either Start or Stop.
• **Start**: launch an Amazon EMR cluster.
• **Stop**: terminate an Amazon EMR cluster. |
<table>
<thead>
<tr>
<th>Region</th>
<th>Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (for example &quot;us-east-1&quot;). For more information about how to specify the AWS region, see Choose an AWS Region.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster name</td>
<td>Enter the name of the cluster.</td>
</tr>
<tr>
<td>Cluster version</td>
<td>Select the version of the cluster. You can also select the Custom Version and Application check box on the Advanced settings view to customize the cluster version information. This property is not available when the Customize Version and Application check box is selected.</td>
</tr>
<tr>
<td>Application</td>
<td>Select the applications to be installed on the cluster. You can also select the Custom Version and Application check box on the Advanced settings view to customize the applications information. This property is available when an EMR version is selected from the Cluster version list and the Customize Version and Application check box is cleared.</td>
</tr>
<tr>
<td>Service role</td>
<td>Enter the IAM (Identity and Access Management) role for the Amazon EMR service. The default role is EMR_DefaultRole. To use this default role, you must have already created it.</td>
</tr>
<tr>
<td>Job flow role</td>
<td>Enter the IAM role for the EC2 instances that Amazon EMR manages. The default role is EMR_EC2_DefaultRole. To use this default role, you must have already created it.</td>
</tr>
<tr>
<td>Enable log</td>
<td>Select this check box to enable logging and in the field displayed specify the path to a folder in an S3 bucket where you want Amazon EMR to write the log data.</td>
</tr>
<tr>
<td>Use EC2 key pair</td>
<td>Select this check box to associate an Amazon EC2 (Elastic Compute Cloud) key pair with the cluster and in the field displayed enter the name of your EC2 key pair.</td>
</tr>
</tbody>
</table>
| Predicate | Specify the cluster(s) that you want to stop:  
  - All running clusters: all running clusters will be stopped.  
  - All running clusters with predefined name: the running cluster with a given name will be stopped. In the Cluster name field displayed, you need to specify the name of the cluster to be stopped.  
  - Running cluster with predefined id: the running cluster with a given ID will be stopped. In the Cluster id field displayed, you need to specify the ID of the cluster to be stopped.  
This list is available only when Stop is selected from the Action list. |
<table>
<thead>
<tr>
<th><strong>Instance count</strong></th>
<th>Enter the number of Amazon EC2 instances to initialize.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Master instance type</strong></td>
<td>Select the type of the master instance to initialize.</td>
</tr>
<tr>
<td><strong>Slave instance type</strong></td>
<td>Select the type of the slave instance to initialize.</td>
</tr>
</tbody>
</table>

### Advanced settings

| **STS Endpoint** | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from. This check box is available only when the Assume role check box is selected. |
| **Wait for cluster ready** | Select this check box to let your Job wait until the launch of the cluster is completed. |
| **Visible to all users** | Select this check box to make the cluster visible to all IAM users. |
| **Termination Protect** | Select this check box to enable termination protection to prevent instances in the cluster from shutting down due to errors or issues during processing. |
| **Enable debug** | Select this check box to enable the debug mode. |

### Customize Version and Application

Select this check box to customize the version of the cluster and the applications to be installed on the cluster.

- **Cluster version**: enter the version of the cluster.
- **Applications**: click the [+] button below the table to add as many rows as needed, each row for an application, and specify the application by clicking the right side of the cell and selecting the application from the drop-down list displayed, or just entering the application name in the cell if it is not in the list.

| **Subnet id** | Specify the identifier of the Amazon VPC (Virtual Private Cloud) subnet where you want the job flow to launch. |
| **Availability Zone** | Specify the availability zone for your cluster’s EC2 instances. |
| **Master security group** | Specify the security group for the master instance. |
| **Additional master security groups** | Specify additional security groups for the master instance and separate them with a comma, for example, gname1, gname2, gname3. |
| **Slave security group** | Specify the security group for the slave instances. |
| **Additional slave security groups** | Specify additional security groups for the slave instances and separate them with a comma, for example, gname1, gname2, gname3. |
### Service Access Security Group
Specify the identifier of the Amazon EC2 security group for the Amazon EMR service to access clusters in VPC private subnet.

For how to create a private subnet to enable service access security group on Amazon EMR, see Scenario 2: VPC with Public and Private Subnets (NAT).

### Actions
Specify the bootstrap actions associated with the cluster, by clicking the [+ ] button below the table to add as many rows as needed, each row for a bootstrap action, and setting the following parameters for each action:

- **Name**: enter the name of the bootstrap action.
- **Script location**: specify the location of the script run by the bootstrap action, for example, `s3://ap-northeast-1.elasticmapreduce/bootstrap-actions/run-if`.
- **Arguments**: enter the list of command line arguments (separated by commas) passed to the bootstrap action script, for example, `",arg0","arg1","arg2"`. For more information about the bootstrap actions, see BootstrapActionConfig.

### Steps
Specify the job flow step(s) to be invoked on the cluster after its launch, by clicking the [+ ] button below the table to add as many rows as needed, each row for a step, and setting the following parameters for each step:

- **Name**: enter the name of the job flow step.
- **Action on Failure**: click the cell and from the drop-down list select the action to take if the job flow step fails.
- **Main Class**: enter the name of the main class in the specified Java file. If not specified, the JAR file should specify a Main-Class in its manifest file.
- **Jar**: enter the path to the JAR file run during the step, for example, `s3://inputjar/test.jar`.
- **Args**: enter the list of command line arguments (separated by commas) passed to the JAR file’s main function when executed, for example, `",arg0","arg1","arg2"`. For more information about the job flow steps, see StepConfig.

### Keep alive after steps complete
Select this check box to keep the job flow alive after completing all steps.

### Wait for steps to complete
Select this check box to let your Job wait until the job flow steps are completed.

This check box is available only when the Wait for cluster ready check box is selected.

### Properties
Specify the classification and property information supplied to the configuration object of the EMR cluster.
to be created, by clicking the [+] button below the table to add as many rows as needed, each row for a property, and setting the following parameters:

- **Classification**: specify the classification of the configuration.
- **Key**: enter the key of the property.
- **Value**: enter the value of the property.

### tStatCatcher Statistics
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLUSTER_FINAL_ID</td>
<td>The ID of the cluster. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>CLUSTER_FINAL_NAME</td>
<td>The name of the cluster. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

### Usage

#### Usage rule
**tAmazonEMRManage** is usually used as a standalone component.

### Managing an Amazon EMR cluster

Here’s an example of using Talend components to manage an Amazon EMR cluster.

#### Creating an Amazon EMR cluster management Job

Create a Job to start a new Amazon EMR cluster, then resize the cluster, and finally list the ID and name information of the instance groups in the cluster.
Procedure

1. Create a new Job and add a **tAmazonEMRManage** component, a **tAmazonEMRResize** component, a **tAmazonEMRListInstances** component, and a **tJava** component by typing their names in the design workspace or dropping them from the **Palette**.

2. Link the **tAmazonEMRManage** component to the **tAmazonEMRResize** component using a **Trigger > OnSubjobOk** connection.

3. Link the **tAmazonEMRResize** component to the **tAmazonEMRListInstances** component using a **Trigger > OnSubjobOk** connection.

4. Link the **tAmazonEMRListInstances** component to the **tJava** component using a **Row > Iterate** connection.

**Starting a new Amazon EMR cluster**

Configure the **tAmazonEMRManage** component to start a new Amazon EMR cluster.

Procedure

1. Double-click the **tAmazonEMRManage** component to open its **Basic settings** view.
2. In the **Access Key** and **Secret Key** fields, enter the authentication credentials required to access Amazon S3.

3. From the **Action** list, select **Start** to start a cluster.

4. Select the AWS region from the **Region** drop-down list. In this example, it is *Asia Pacific (Tokyo)*.

5. In the **Cluster name** field, enter the name of the cluster to be started. In this example, it is *talend-doc-emr-cluster*.

6. From the **Cluster version** and **Application** drop-down list, select the version of the cluster and the application to be installed on the cluster.

7. Select the **Enable log** check box and in the field displayed, specify the path to a folder in an S3 bucket where you want Amazon EMR to write the log data. In this example, it is *s3://talend-doc-emr-bucket*.

### Resizing the Amazon EMR cluster by adding a new task instance group

Configure the **tAmazonEMRResize** component to resize a running Amazon EMR cluster by adding a new task instance group.

**Procedure**

1. Double-click the **tAmazonEMRResize** component to open its **Basic settings** view.
2. In the Access Key and Secret Key fields, enter the authentication credentials required to access Amazon S3.

3. From the Action drop-down list, select Add task instance group to resize the cluster by adding a new task instance group.

4. In the Cluster id field, enter the ID of the cluster to be resized. In this example, the returned value of the global variable CLUSTER_FINAL_ID of the previous tAmazonEMRManage component is used.

   Note that you can retrieve the global variable by pressing Ctrl + Space and selecting the relevant global variable from the list.

5. In the Group name field, enter the name of the task instance group to be added in the cluster. In this example, it is talend-doc-instance-group.

6. In the Instance count field, specify the number of the instances to be created.

7. From the Task instance type drop-down list, select the type of the instances to be created.

## Listing the instance groups in the Amazon EMR cluster

Configure the tAmazonEMRListInstances component and the tJava component to retrieve and display the ID and name information of all instance groups in a running cluster.

### Procedure

1. Double-click the tAmazonEMRListInstances component to open its Basic settings view.

2. In the Access Key and Secret Key fields, enter the authentication credentials required to access Amazon S3.

3. Select the AWS region from the Region drop-down list. In this example, it is Asia Pacific (Tokyo).

4. Clear the Filter master and core instances check box to list all instance groups, including the Master, Core, and Task type instance groups.

5. In the Cluster id field, enter the ID of the cluster for which to list the instance groups. In this example, the returned value of the global variable CLUSTER_FINAL_ID of the previous tAmazonEMRManage component is used.

6. Double-click the tJava component to open its Basic settings view.
7. In the **Code** field, enter the following code to print the ID and Name information of each instance group in the cluster.

```java
System.out.println("\r\n===== Instance Group =====");
System.out.println("Instance Group ID:    " + (String)globalMap.get("tAmazonEMRListInstances_1_CURRENT_GROUP_ID");
System.out.println("Instance Group Name:  " + (String)globalMap.get("tAmazonEMRListInstances_1_CURRENT_GROUP_NAME");
```

**Executing the Job to manage the Amazon EMR cluster**

After setting up the Job and configuring the components used in the Job for managing Amazon EMR cluster, you can then execute the Job and verify the Job execution result.

**Procedure**

1. Press **Ctrl + S** to save the Job and then **F6** to execute the Job.

   ```none
   Starting job DOC77011_tAmazonEMR XXXX at 16:50 24-09-2016.
   [statistics] connecting to socket on port 3612
   [statistics] connected
   Waiting for cluster to become available.
   ..........................................................
   ..........................................................
   ..........................................................
   ..........................................................
   ---- Instance Group ----
   Instance Group ID:  ig-3DF4Y11W2EZV1Y
   Instance Group Name:  master
   ---- Instance Group ----
   Instance Group ID:  ig-17BB2U9C14KX2
   Instance Group Name:  talend-doc-instance-group
   ---- Instance Group ----
   Instance Group ID:  ig-3HL9UDW30CD33
   Instance Group Name:  slave
   [statistics] disconnected
   Job DOC77011_tAmazonEMR XXXX ended at 17:00 24-09-2016. [exit code=0]
   ```

   As shown above, the Job starts and resizes the Amazon EMR cluster, and then lists all instance groups in the cluster.

2. View the cluster details on the Amazon EMR Cluster List page to validate the Job execution result.
tAmazonEMRResize

Adds or resizes a task instance group in a cluster on Amazon EMR (Elastic MapReduce).

**tAmazonEMRResize Standard properties**

These properties are used to configure tAmazonEMRResize running in the Standard Job framework.

The Standard tAmazonEMRResize component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

| **Access key** and **Secret key** | Specify the access keys (the access key ID in the **Access Key** field and the secret access key in the **Secret Key** field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key).

To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Inherit credentials from AWS role** | Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances. |
| **Assume role** | Select this check box and specify the values for the following parameters used to create a new assumed role session.

- **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.
- **Role session name**: an identifier for the assumed role session.
- **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.

For more information on assuming roles, see AssumeRole. |
| **Action** | Select an action to be performed from the drop-down list.

- **Add task instance group**: add a task instance group in a cluster.
- **Resize task instance group**: resize a task instance group in a cluster. |
| Region | Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (for example 'us-east-1'). For more information about how to specify the AWS region, see Choose an AWS Region. |
| Cluster id | Enter the ID of the cluster to be resized. |
| Group name | Enter the name of the task instance group to be added. This field is available only when **Add task instance group** is selected from the **Action** drop-down list. |
| Group id | Enter the ID of the task instance group to be resized. This field is available only when **Resize task instance group** is selected from the **Action** drop-down list. |
| Instance count | Enter the number of instances for the task instance group. |
| Task instance type | Select an instance type for all instances in the task instance group to be added from the drop-down list. This list is available only when **Add task instance group** is selected from the **Action** drop-down list. |
| Request spot | Select this check box to launch Spot instances, and in the **Bid price($)** field displayed, enter the maximum hourly rate (in dollars) you are willing to pay per instance. This check box is available only when **Add task instance group** is selected from the **Action** drop-down list. |

**Advanced settings**

| STS Endpoint | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from. This check box is available only when the **Assume role** check box is selected. |
| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | **TASK_GROUP_ID**: the ID of the task instance group. This is an After variable and it returns a string.  
**TASK_GROUP_NAME**: the name of the task instance group. This is an After variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions |
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tAmazonEMRResize is usually used as a standalone component.</th>
</tr>
</thead>
</table>

### Related scenario

No scenario is available for the Standard version of this component yet.
**tAmazonMysqlClose**

Closes the transaction committed in the connected DB.

**tAmazonMysqlClose Standard properties**

These properties are used to configure tAmazonMysqlClose running in the Standard Job framework. The Standard tAmazonMysqlClose component belongs to the Cloud and the Databases families. The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tAmazonMysqlConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

...
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with AmazonMysql components, especially with tAmazonMysqlConnection and tAmazonMysqlCommit.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Related scenarios

No scenario is available for the Standard version of this component yet.
tAmazonMysqlCommit

Commits in one go a global transaction instead of doing that on every row or every batch, and provides gain in performance, using a unique connection.
tAmazonMysqlCommit validates the data processed through the Job into the connected database.

tAmazonMysqlCommit Standard properties

These properties are used to configure tAmazonMysqlCommit running in the Standard Job framework.
The Standard tAmazonMysqlCommit component belongs to the Cloud and the Databases families.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tAmazonMysqlConnection component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

Warning:
If you want to use a Row > Main connection to link tAmazonMysqlCommit to your job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.

Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions |
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | This component is more commonly used with other `tAmazonMysql` components, especially with the `tAmazonMysqlConnection` and `tAmazonMysqlRollback` components. |
| Dynamic settings | Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |

### Related scenario

For `tAmazonMysqlCommit` related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476.
tAmazonMysqlConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

tAmazonMysqlConnection opens a connection to the database for a current transaction.

**tAmazonMysqlConnection Standard properties**

These properties are used to configure tAmazonMysqlConnection running in the Standard Job framework.

The Standard tAmazonMysqlConnection component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>DB Version</td>
<td>MySQL 5 is available.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Additional JDBC parameters</td>
<td>Specify additional connection properties for the DB connection you are creating.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td>Use or register a shared DB Connection</td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the <strong>Shared DB Connection Name</strong> field.</td>
</tr>
</tbody>
</table>
displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the **tRunJob** component. Using a shared connection together with a **tRunJob** component with either of these two options enabled will cause your Job to fail.

### Advanced settings

<table>
<thead>
<tr>
<th>Auto Commit</th>
<th>Select this check box to commit any changes to the database automatically upon the transaction. With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component. Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

**Usage rule**

This component is more commonly used with other **tAmazonMysql** components, especially with the
Related scenario

For a related scenario using this component, see "Scenario: Inserting data in mother/daughter tables" on page 2476.
tAmazonMysqlInput

Reads a database and extracts fields based on a query.

tAmazonMysqlInput executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Row > Main link.

**tAmazonMysqlInput Standard properties**

These properties are used to configure tAmazonMysqlInput running in the Standard Job framework.
The Standard tAmazonMysqlInput component belongs to the Cloud and the Databases families.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>DB Version</strong></td>
<td>MySQL 5 is available.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

**Host**
Database server IP address.

**Port**
Listening port number of DB server.

**Database**
Name of the database.

**Username and Password**
DB user authentication data.
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**Schema and Edit Schema**
A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**.

- **Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.
- **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

**Table Name**
Name of the table to be read.

**Query type and Query**
Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

**Advanced settings**

**Additional JDBC parameters**
Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the **Basic settings**.

- **Note:**
  When you need to handle data of the timestamp type 0000-00-00 00:00:00 using this component, set the parameter as:
  
  noDatetimeStringSync=true&zeroDateTimeBehavior=convertToNull.

**Enable stream**
Select this check box to enable streaming over buffering which allows the code to read from a large table without consuming a large amount of memory in order to optimize the performance.

**Trim all the String/Char columns**
Select this check box to remove leading and trailing whitespace from all the String/Char columns.
| Trim column | Remove leading and trailing whitespace from defined columns.  
  | Note:  
  Deselect Trim all the String/Char columns to enable Trim columns in this field. |
| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
  QUERY: the query statement being processed. This is a Flow variable and it returns a string.  
  ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
  A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
  To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
  For further information about variables, see Talend Studio User Guide. |

**Usage**

| Usage rule | This component covers all possible SQL queries for Mysql databases. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  
  The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.  
  For examples on using dynamic parameters, see Scenario: Reading data from databases through context- |
Related scenarios

For related scenarios, see tMysqlInput on page 2488.
tAmazonMysqlOutput

Writes, updates, makes changes or suppresses entries in a database.

tAmazonMysqlOutput executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tAmazonMysqlOutput Standard properties

These properties are used to configure tAmazonMysqlOutput running in the Standard Job framework. The Standard tAmazonMysqlOutput component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in:</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository:</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>DB Version</td>
<td>MySQL 5 is available.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

Note:
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
</tbody>
</table>

**Action on table**

- **None**: No operation is carried out.
- **Drop and create table**: The table is removed and created again.
- **Create table**: The table does not exist and gets created.
- **Create table if not exists**: The table is created if it does not exist.
- **Drop table if exists and create**: The table is removed if it already exists and created again.
- **Clear table**: The table content is deleted.
- **Truncate table**: The table content is quickly deleted. However, you will not be able to rollback the operation.

**Action on data**

- **Insert**: Add new entries to the table. If duplicates are found, the job stops.
- **Update**: Make changes to existing entries.
- **Insert or update**: Insert a new record. If the record with the given reference already exists, an update would be made.
- **Update or insert**: Update the record with the given reference. If the record does not exist, a new record would be inserted.
- **Delete**: Remove entries corresponding to the input flow.
- **Replace**: Add new entries to the table. If an old row in the table has the same value as a new row for a PRIMARY KEY or a UNIQUE index, the old row is deleted before the new row is inserted.
- **Insert or update on duplicate key or unique index**: Add entries if the inserted value does not exist or update entries if the inserted value already exists and there is a risk of violating a unique index or primary key.
**Insert Ignore**: Add only new rows to prevent duplicate key errors.

**Warning:**

You must specify at least one column as a primary key on which the **Update** and **Delete** operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the update and delete operations. To do that: Select the **Use field options** check box and then in the **Key in update** column, select the check boxes next to the column name on which you want to base the update operation. Do the same in the **Key in delete** column for the deletion operation.

### Schema and Edit schema

A schema is a row description, i.e. it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**.

**Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### Die on error

This check box is selected by default. Clear the check box to skip the row in error and complete the process for error-free rows. If needed, you can retrieve the rows in error via a **Row > Rejects** link.

### Advanced settings

#### Additional JDBC parameters

Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the **Basic settings**.

**Note:**

You can press **Ctrl+Space** to access a list of predefined global variables.
| **Extend Insert** | Select this check box to carry out a bulk insert of a defined set of lines instead of inserting lines one by one. The gain in system performance is considerable. **Number of rows per insert:** enter the number of rows to be inserted per operation. Note that the higher the value specified, the lower performance levels shall be due to the increase in memory demands.  

**Note:** This option is not compatible with the **Reject** link. You should therefore clear the check box if you are using a **Row > Rejects** link with this component.  

**Warning:** If you are using this component with **tMysqlLastInsertID**, ensure that the **Extend Insert** check box in Advanced Settings is not selected. **Extend Insert** allows for batch loading, however, if the check box is selected, only the ID of the last line of the last batch will be returned. |
| **Use Batch** | Select this check box to activate the batch mode for data processing.  

**Note:** This check box is available only when you have selected the **Update** or the **Delete** option in the **Action on data** field. |
| **Batch Size** | Specify the number of records to be processed in each batch. This field appears only when the **Use batch mode** check box is selected. |
| **Commit every** | Number of rows to be included in the batch before it is committed to the DB. This option ensures transaction quality (but not rollback) and, above all, a higher performance level. |
| **Additional Columns** | This option is not available if you have just created the DB table (even if you delete it beforehand). This option allows you to call SQL functions to perform actions on columns, provided that these are not insert, update or delete actions, or actions that require pre-processing.  

**Name:** Type in the name of the schema column to be altered or inserted.  

**SQL expression:** Type in the SQL statement to be executed in order to alter or insert the data in the corresponding column.  

**Position:** Select **Before**, **Replace** or **After**, depending on the action to be performed on the reference column. |
| Reference column: Type in a reference column that tAmazonMysqlOutput can use to locate or replace the new column, or the column to be modified. |
| **Use field options** | Select this check box to customize a request, particularly if multiple actions are being carried out on the data. |
| **Use Hint Options** | Select this check box to activate the hint configuration area which helps you optimize a query’s execution. In this area, parameters are: |
|  | - **HINT**: specify the hint you need, using the syntax /*+ */. |
|  | - **POSITION**: specify where you put the hint in a SQL statement. |
|  | - **SQL STMT**: select the SQL statement you need to use. |
| **Enable debug mode** | Select this check box to display each step involved in the process of writing data in the database. |
| **Use duplicate key update mode insert** | Updates the values of the columns specified, in the event of duplicate primary keys. |
|  | **Column**: Between double quotation marks, enter the name of the column to be updated. |
|  | **Value**: Enter the action you want to carry out on the column. |
|  | **Note:** To use this option you must first of all select the **Insert** mode in the **Action on data** list found in the **Basic Settings** view. |
| **tStatCatcher Statistics** | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. |
|  | **NB_LINE_UPDATED**: the number of rows updated. This is an After variable and it returns an integer. |
|  | **NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer. |
|  | **NB_LINE_DELETED**: the number of rows deleted. This is an After variable and it returns an integer. |
|  | **NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer. |
|  | **QUERY**: the query statement processed. This is an After variable and it returns a string. |
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a MySQL database. It also allows you to create a reject flow using a Row > Rejects link to filter data in error. For an example of tAmazonMysqlOutput in use, see .

Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

For related scenarios, see tMysqlSCD on page 2563.
tAmazonMysqlRollback

Cancels the transaction commit in the connected database and avoids to commit part of a transaction involuntarily.

**tAmazonMysqlRollback Standard properties**

These properties are used to configure tAmazonMysqlRollback running in the Standard Job framework.

The Standard tAmazonMysqlRollback component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tAmazonMysqlConnection component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

### Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

### Global Variables

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
Usage

Usage rule

This component is more commonly used with other tAmazonMysql* components, especially with the tAmazonMysqlConnection and tAmazonMysqlCommit components.

Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenario

For a related scenario, see Scenario: Rollback from inserting data in mother/daughter tables on page 2480.
tAmazonMysqlRow

Executes the SQL query stated onto the specified database.

Depending on the nature of the query and the database, tAmazonMysqlRow acts on the actual DB structure or on the data (although without handling data). The SQLBuilder tool helps you write easily your SQL statements. tAmazonMysqlRow is the specific component for this database query. The row suffix means the component implements a flow in the job design although it does not provide output.

tAmazonMysqlRow Standard properties

These properties are used to configure tAmazonMysqlRow running in the Standard Job framework. The Standard tAmazonMysqlRow component belongs to the Cloud and the Databases families. The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>DB Version</td>
<td>MySQL 5 is available.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>Note: When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
<td></td>
</tr>
<tr>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
<td></td>
</tr>
</tbody>
</table>
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Database server IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description, that is to say, it defines the number of fields to be processed and passed on to the next component. The schema is either <strong>Built-in</strong> or stored remotely in the <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>. When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
<tr>
<td>Table Name</td>
<td>Name of the table to be processed.</td>
</tr>
<tr>
<td>Query type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>Fill in manually the query statement or build it graphically using SQLBuilder.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</td>
</tr>
<tr>
<td>Guess Query</td>
<td>Click the <strong>Guess Query</strong> button to generate the query which corresponds to your table schema in the <strong>Query</strong> field.</td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>

### Advanced settings

| **Additional JDBC parameters** | Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the **Basic settings**. |
| **Propagate QUERY's recordset** | Select this check box to insert the result of the query in a COLUMN of the current flow. Select this column from the **use column** list.  

**Note:**  
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY's recordset should be set to the type of Object and this component is usually followed by **tParseRecordSet**. |
| **Use PreparedStatement** | Select this check box if you want to query the database using a PreparedStatement. In the **Set PreparedStatement Parameter** table, define the parameters represented by '?' in the SQL instruction of the **Query** field in the **Basic Settings** tab.  
**Parameter Index:** Enter the parameter position in the SQL instruction.  
**Parameter Type:** Enter the parameter type.  
**Parameter Value:** Enter the parameter value.  

**Note:**  
This option is very useful if you need to execute the same query several times. Performance levels are increased |
| **Commit every** | Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions. |
| **tStatCatcher Statistics** | Select this check box to collect log data at the component level. |

### Global Variables

| **Global Variables** | **QUERY**: the query statement being processed. This is a Flow variable and it returns a string. |
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

---

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
</table>

**Dynamic settings**

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections on page 2497* and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489*. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

---

**Related scenario**

For a related scenario, see:

- *Scenario: Combining two flows for selective output on page 2558*
tAmazonOracleClose

Closes the transaction committed in the connected database.

tAmazonOracleClose Standard properties

These properties are used to configure tAmazonOracleClose running in the Standard Job framework. The Standard tAmazonOracleClose component belongs to the Cloud and the Databases families. The component in this framework is generally available.

Note:
- This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
- For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tAmazonOracleConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |


Usage

| Usage rule | This component is to be used along with AmazonOracle components, especially with tAmazonOracleConnection and tAmazonOracleCommit. |
| Dynamic settings | Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

Related scenario

This component is to be used with tAmazonOracleConnection and tAmazonOracleRollback components. It is generally used with a tAmazonOracleConnection to close a connection for the ongoing transaction.

For a related scenario, see tMysqlConnection on page 2475.
tAmazonOracleCommit

Commits in one go a global transaction instead of doing that on every row or every batch, and provides gain in performance, using a unique connection.

tAmazonOracleCommit validates the data processed through the Job into the connected database.

**tAmazonOracleCommit Standard properties**

These properties are used to configure tAmazonOracleCommit running in the Standard Job framework.

The Standard tAmazonOracleCommit component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

ℹ️ **Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tAmazonOracleConnection</strong> component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

⚠️ **Warning:**

*If you want to use a Row > Main connection to link tAmazonOracleCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.*

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions |
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other tAmazonOracle* components, especially with the tAmazonOracleConnection and tAmazonOracleRollback components.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [+ ] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |

### Related scenario

For **tAmazonOracleCommit** related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476
tAmazonOracleConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.
tAmazonOracleConnection opens a connection to the database for a current transaction.

tAmazonOracleConnection Standard properties

These properties are used to configure tAmazonOracleConnection running in the Standard Job framework.

The Standard tAmazonOracleConnection component belongs to the Cloud and the Databases families.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Connection type</td>
<td>Drop-down list of available drivers:</td>
</tr>
<tr>
<td>Oracle SID</td>
<td>Select this connection type to uniquely identify a particular database on a system.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Oracle 11-5 is available.</td>
</tr>
<tr>
<td>Use tns file</td>
<td>Select this check box to use the metadata of a context included in a tns file.</td>
</tr>
</tbody>
</table>

Note:
One tns file may have many contexts.

TNS File: Enter the path to the tns file manually or browse to the file by clicking the three-dot button next to the file.

Select a DB Connection in Tns File: Click the three-dot button to display all the contexts held in the tns file and select the desired one.
<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Additional JDBC parameters</strong></td>
<td>Specify additional connection properties for the DB connection you are creating. <strong>Note:</strong> You can set the encoding parameters through this field.</td>
</tr>
<tr>
<td><strong>Use or register a shared DB Connection</strong></td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the <strong>Shared DB Connection Name</strong> field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child. This option is incompatible with the <strong>Use dynamic job</strong> and <strong>Use an independent process to run subjob</strong> options of the <strong>tRunJob</strong> component. Using a shared connection together with a <strong>tRunJob</strong> component with either of these two options enabled will cause your Job to fail.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **Auto Commit** | Select this check box to commit any changes to the database automatically upon the transaction. With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component. Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component. |
tStatCatcher Statistics

Select this check box to gather the job processing metadata at a job level as well as at each component level.

Global Variables

Global Variables

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

This component is more commonly used with other tAmazonOracle* components, especially with the tAmazonOracleCommit and tAmazonOracleRollback components.

Related scenario

For tAmazonOracleConnection related scenario, see tMysqlConnection on page 2475
**tAmazonOracleInput**

Reads a database and extracts fields based on a query.

tAmazonOracleInput executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a **Row > Main** link.

**tAmazonOracleInput Standard properties**

These properties are used to configure tAmazonOracleInput running in the Standard Job framework. The Standard tAmazonOracleInput component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Connection type</strong></td>
<td>Drop-down list of available drivers:</td>
</tr>
<tr>
<td></td>
<td><strong>Oracle SID</strong>: Select this connection type to uniquely identify a particular database on a system.</td>
</tr>
<tr>
<td><strong>DB Version</strong></td>
<td>Select the Oracle version in use.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Oracle schema</td>
<td>Oracle schema name.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description, i.e. it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository.</td>
</tr>
<tr>
<td>Built-in</td>
<td>The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Repository</td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Table name</td>
<td>Database table name.</td>
</tr>
<tr>
<td>Query type and Query</td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use cursor</td>
<td>When selected, helps to decide the row set to work with at a time and thus optimize performance.</td>
</tr>
<tr>
<td>Trim all the String/Char columns</td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</td>
</tr>
<tr>
<td>Trim column</td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
</tbody>
</table>
### Global Variables

| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |

### Usage

| Usage rule | This component covers all possible SQL queries for Oracle databases. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  
The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.  
For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*. |

### Limitation

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the |
Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Related scenarios

For related scenarios, see:

- Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
tAmazonOracleOutput

Writes, updates, makes changes or suppresses entries in a database.

tAmazonOracleOutput executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

**tAmazonOracleOutput Standard properties**

These properties are used to configure tAmazonOracleOutput running in the Standard Job framework.
The Standard tAmazonOracleOutput component belongs to the Cloud and the Databases families.
The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide.*

<table>
<thead>
<tr>
<th>Connection type</th>
<th>Drop-down list of available drivers: <strong>Oracle SID</strong>: Select this connection type to uniquely identify a particular database on a system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Version</td>
<td>Select the Oracle version in use.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Oracle schema</td>
<td>Name of the Oracle schema.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations: <strong>None</strong>: No operation is carried out. <strong>Drop and create a table</strong>: The table is removed and created again. <strong>Create a table</strong>: The table does not exist and gets created. <strong>Create a table if not exists</strong>: The table is created if it does not exist. <strong>Drop a table if exists and create</strong>: The table is removed if it already exists and created again. <strong>Clear a table</strong>: The table content is deleted. <strong>Warning</strong>: If you select the <strong>Use an existing connection</strong> check box and select an option other than <strong>None</strong> from the <strong>Action on table</strong> list, a commit statement will be generated automatically before the data update/insert/delete operation.</td>
</tr>
<tr>
<td>Action on data</td>
<td>On the data of the table defined, you can perform: <strong>Insert</strong>: Add new entries to the table. If duplicates are found, job stops. <strong>Update</strong>: Make changes to existing entries <strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
</tbody>
</table>
**Update or insert**: Update the record with the given reference. If the record does not exist, a new record would be inserted.

**Delete**: Remove entries corresponding to the input flow.

**Warning:**

It is necessary to specify at least one column as a primary key on which the **Update** and **Delete** operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the **Update** and **Delete** operations. To do that: Select the **Use field options** check box and then in the **Key in update column**, select the check boxes next to the column names you want to use as a base for the **Update** operation. Do the same in the **Key in delete column** for the **Delete** operation.

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>A schema is a row description, i.e. it defines the number of fields to be processed and passed on to the next component. The schema is either <strong>Built-in</strong> or stored remotely in the <strong>Repository</strong>.</td>
</tr>
</tbody>
</table>

| Built-In: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**. |

<table>
<thead>
<tr>
<th>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <strong>Talend Studio User Guide</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.</td>
</tr>
<tr>
<td>You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Die on error</th>
</tr>
</thead>
<tbody>
<tr>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Additional JDBC parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <strong>Use an existing connection</strong> check box in the <strong>Basic settings</strong>.</td>
</tr>
</tbody>
</table>

**Note:**
You can press **Ctrl+Space** to access a list of predefined global variables.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Override any existing NLS_LANG environment variable</td>
<td>Select this check box to override variables already set for a NLS language environment.</td>
</tr>
<tr>
<td>Commit every</td>
<td>Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.</td>
</tr>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
<tr>
<td>Additional Columns</td>
<td>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</td>
</tr>
<tr>
<td>Name</td>
<td>Type in the name of the schema column to be altered or inserted as new column.</td>
</tr>
<tr>
<td>SQL expression</td>
<td>Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td>Position</td>
<td>Select Before, Replace or After following the action to be performed on the reference column.</td>
</tr>
<tr>
<td>Reference column</td>
<td>Type in a column of reference that the tDBOutput can use to place or replace the new or altered column.</td>
</tr>
<tr>
<td>Use field options</td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td>Use Hint Options</td>
<td>Select this check box to activate the hint configuration area which helps you optimize a query’s execution. In this area, parameters are:</td>
</tr>
<tr>
<td>HINT</td>
<td>specify the hint you need, using the syntax /*+ */.</td>
</tr>
<tr>
<td>POSITION</td>
<td>specify where you put the hint in a SQL statement.</td>
</tr>
<tr>
<td>SQL STMT</td>
<td>select the SQL statement you need to use.</td>
</tr>
<tr>
<td>Convert columns and table to uppercase</td>
<td>Select this check box to set the names of columns and table in uppercase.</td>
</tr>
<tr>
<td>Enable debug mode</td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td>Use Batch</td>
<td>Select this check box to activate the batch mode for data processing.</td>
</tr>
</tbody>
</table>
### Batch Size

Specify the number of records to be processed in each batch.

This field appears only when the **Use batch mode** check box is selected.

### Support null in “SQL WHERE” statement

Select this check box to validate null in “SQL WHERE” statement.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_LINE_UPDATED: the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_INSERTED: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_DELETED: the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_REJECTED: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>QUERY: the query statement processed. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

#### Usage rule

This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in an Oracle database. It also allows you to create a reject flow using a **Row > Rejects** link to filter data in error. For such an example, see *Scenario: Retrieving data in error with a Reject link* on page 2529.

#### Dynamic settings

Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple
connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

### Related scenarios

For **tAmazonOracleOutput** related topics, see:

- **Scenario: Inserting a column and altering data using tMysqlOutput** on page 2521.
tAmazonOracleRollback

Cancels the transaction commit in the connected database and avoids to commit part of a transaction involuntarily.

tAmazonOracleRollback Standard properties

These properties are used to configure tAmazonOracleRollback running in the Standard Job framework.

The Standard tAmazonOracleRollback component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tAmazonOracleConnection component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. |

For further information about variables, see *Talend Studio User Guide*.

### Usage

**Usage rule**

This component is more commonly used with other *tAmazonOracle* components, especially with the *tAmazonOracleConnection* and *tAmazonOracleCommit* components.

**Dynamic settings**

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Related scenario

For *tAmazonOracleRollback* related scenario, see *tMysqlRollback* on page 2546.
tAmazonOracleRow

Executes the SQL query stated onto the specified database.

Depending on the nature of the query and the database, tAmazonOracleRow acts on the actual DB structure or on the data (although without handling data). The SQLBuilder tool helps you write easily your SQL statements. tAmazonOracleRow is the specific component for this database query. The row suffix means the component implements a flow in the job design although it does not provide output.

**tAmazonOracleRow Standard properties**

These properties are used to configure tAmazonOracleRow running in the Standard Job framework. The Standard tAmazonOracleRow component belongs to the Cloud and the Databases families. The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Connection type</strong></th>
<th>Drop-down list of available drivers.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
</tbody>
</table>
| **Username and Password** | DB user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Schema and Edit Schema** | A schema is a row description, i.e. it defines the number of fields to be processed and passed on to the next component. The schema is either *Built-in* or stored remotely in the *Repository*. |
| **Query type**      | Either *Built-in* or *Repository*. |
| **Query**           | Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition. |
| **Use NB_LINE_**    | This option allows you feed the variable with the number of rows inserted/updated/deleted to the next component or subjob. This field only applies if the query entered in *Query* field is a *INSERT*, *UPDATE* or *DELETE* query.  
• **NONE**: does not feed the variable.  
• **INSERTED**: feeds the variable with the number of rows inserted.  
• **UPDATED**: feeds the variable with the number of rows updated.  
• **DELETED**: feeds the variable with the number of rows deleted. |

*Built-in*: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.  
*Repository*: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.  
*Query type*  
*Built-in*: Fill in manually the query statement or build it graphically using SQLBuilder  
*Repository*: Select the relevant query stored in the Repository. The Query field gets accordingly filled in.  
*Query*  
This option allows you feed the variable with the number of rows inserted/updated/deleted to the next component or subjob. This field only applies if the query entered in *Query* field is a *INSERT*, *UPDATE* or *DELETE* query.  
• **NONE**: does not feed the variable.  
• **INSERTED**: feeds the variable with the number of rows inserted.  
• **UPDATED**: feeds the variable with the number of rows updated.  
• **DELETED**: feeds the variable with the number of rows deleted.
### Advanced settings

| Propagate QUERY’s recordset | Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the use column list.  

*Note:*
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of `Object` and this component is usually followed by `tParseRecordSet`. |
|---|---|
| Use PreparedStatement | Select this check box if you want to query the database using a PreparedStatement. In the Set PreparedStatement Parameter table, define the parameters represented by `?` in the SQL instruction of the Query field in the Basic Settings tab.  

**Parameter Index:** Enter the parameter position in the SQL instruction.  

**Parameter Type:** Enter the parameter type.  

**Parameter Value:** Enter the parameter value.  

*Note:*  
This option is very useful if you need to execute the same query several times. Performance levels are increased |
| Commit every | Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions. |
| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

### Global Variables

| Global Variables | **QUERY:** the query statement being processed. This is a Flow variable and it returns a string.  

**NB_LINE_UPDATED:** the number of rows updated. This is an After variable and it returns an integer.  

**NB_LINE_INSERTED:** the number of rows inserted. This is an After variable and it returns an integer.  

**NB_LINE_DELETED:** the number of rows deleted. This is an After variable and it returns an integer. |

| Die on error | This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row > Repeats link. |
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

---

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
</table>

**Dynamic settings**

Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

---

**Related scenarios**

For related topics, see:

- *Scenario: Combining two flows for selective output* on page 2558
- *Procedure* on page 625.
- *Scenario: Removing and regenerating a MySQL table index* on page 2552.
tAmazonRedshiftManage

Manages Amazon Redshift clusters and snapshots.

tAmazonRedshiftManage manages the work of creating a new Amazon Redshift cluster, creating a snapshot of an Amazon Redshift cluster, resizing an existing Amazon Redshift cluster, and deleting an existing cluster or snapshot.

**tAmazonRedshiftManage Standard properties**

These properties are used to configure tAmazonRedshiftManage running in the Standard Job framework.

The Standard tAmazonRedshiftManage component belongs to the Cloud and the Databases families. The component in this framework is generally available.

**Basic settings**

| Access Key and Secret Key | Specify the access keys (the access key ID in the Access Key field and the secret access key in the Secret Key field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key).

To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Inherit credentials from AWS role | Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances. |
| Assume role | Select this check box and specify the values for the following parameters used to create a new assumed role session.

- **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.
- **Role session name**: an identifier for the assumed role session.
- **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.

For more information on assuming roles, see AssumeRole. |
| Action | Select an action to be performed from the list.

- **Create cluster**: create a new Amazon Redshift cluster. |
- **Delete cluster**: delete a previously provisioned Amazon Redshift cluster.
- **Resize cluster**: resize an existing Amazon Redshift cluster.
- **Restore from snapshot**: create a new Amazon Redshift cluster from a snapshot.
- **Delete snapshot**: delete the specified manual snapshot.

### Region

Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (e.g. `"us-east-1"`) in the list. For more information about the supported AWS regions where you can provision an Amazon Redshift cluster, see [Regions and Endpoints](#).

### Create snapshot

Select this check box to create a final snapshot of the Amazon Redshift cluster before it is deleted.

This check box is available only when **Delete cluster** is selected from the **Action** list.

### Snapshot id

Enter the identifier of the snapshot.

This field is available when:

- **Delete cluster** is selected from the **Action** list and the **Create snapshot** check box is selected.
- **Restore from snapshot** or **Delete snapshot** is selected from the **Action** list.

### Cluster id

Enter the ID of the cluster.

This field is available when **Create cluster**, **Delete cluster**, **Resize cluster**, or **Restore from snapshot** is selected from the **Action** list.

### Database

Enter the name of the first database to be created when the cluster is created.

This field is available only when **Create cluster** is selected from the **Action** list.

### Port

Enter the port number on which the cluster accepts connections.

This field is available when **Create cluster** or **Restore from snapshot** is selected from the **Action** list.

### Master username and Master password

The user name and the password associated with the master user account for the cluster to be created.

To enter the password, click the `[...]` button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

The two fields are available only when **Create cluster** is selected from the **Action** list.
| **Node type** | Select the node type for the cluster.  
This list is available when **Create cluster**, **Resize cluster**, or **Restore from snapshot** is selected from the **Action** list. |
| **Node count** | Enter the number of compute nodes in the cluster.  
This field is available only when **Create cluster** or **Resize cluster** is selected from the **Action** list. |

**Advanced settings**

| **STS Endpoint** | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.  
This check box is available only when the **Assume role** check box is selected. |
| **Wait for cluster ready** | Select this check box to let your Job wait until the launch of the cluster is completed.  
This check box is available when **Create cluster** or **Restore from snapshot** is selected from the **Action** list. |
| **Original cluster id of snapshot** | Enter the name of the cluster the source snapshot was created from.  
This field is available when **Restore from snapshot** or **Delete snapshot** is selected from the **Action** list. |
| **Parameter group name** | Enter the name of the parameter group to be associated with the cluster.  
This field is available when **Create cluster** or **Restore from snapshot** is selected from the **Action** list. |
| **Subnet group name** | Enter the name of the subnet group where you want the cluster to be restored.  
This field is available when **Create cluster** or **Restore from snapshot** is selected from the **Action** list. |
| **Publicly accessible** | Select this check box so that the cluster can be accessed from a public network.  
This check box is available when **Create cluster** or **Restore from snapshot** is selected from the **Action** list. |
| **Set public ip address** | Select this check box and in the field displayed enter the Elastic IP (EIP) address for the cluster.  
This check box is available only when the **Publicly accessible** check box is selected. |
| **Availability zone** | Enter the EC2 Availability Zone in which you want Amazon Redshift to provision the cluster.  
This field is available when **Create cluster** or **Restore from snapshot** is selected from the **Action** list. |
VPC security group ids

Enter Virtual Private Cloud (VPC) security groups to be associated with the cluster and separate them with a comma, for example, `gname1, gname2, gname3`.

This field is available when Create cluster or Restore from snapshot is selected from the Action list.

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

---

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>CLUSTER_FINAL_ID: the ID of the cluster. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ENDPOINT: the endpoint address of the cluster. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

---

**Usage**

| Usage rule | tAmazonRedshiftManage is usually used as a standalone component. |

---

**Related scenario**

No scenario is available for the Standard version of this component yet.
tApacheLogInput

Reads the access-log file for an Apache HTTP server.

To effectively manage the Apache HTTP Server, it is necessary to get feedback about the activity and performance of the server as well as any problems that may be occurring.

**tApacheLogInput Standard properties**

These properties are used to configure tApacheLogInput running in the Standard Job framework.

The Standard tApacheLogInput component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file where the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Schema and Edit Schema**

A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**.

In the context of tApacheLogInput usage, the schema is read-only.

<table>
<thead>
<tr>
<th></th>
<th><strong>Built-in</strong>: You can create the schema and store it locally for this component. Related topic: see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Repository</strong>: You have already created and stored the schema in the Repository. You can reuse it in various projects and Job flowcharts. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

**File Name**

Name of the file and/or the variable to be processed.

For further information about how to define and use a variable in a Job, see Talend Studio User Guide.

**Die on error**

Select this check box to stop the execution of the Job when an error occurs. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can collect the rows on error using a Row > Reject link.
Advanced settings

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Select the encoding type from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for DB data handling.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule | **tApacheLogInput** can be used with other components or as a standalone component. It allows you to create a data flow using a **Row > Main** connection, or to create a reject flow to filter specified data using a **Row > Reject** connection. For an example of how to use these two links, see **Procedure** on page 971. |

Scenario: Reading an Apache access-log file

The following scenario creates a two-component Job, which aims at reading the access-log file for an Apache HTTP server and displaying the output in the **Run** log console.

Procedure

**Procedure**

1. Drop a **tApacheLogInput** component and a **tLogRow** component from the **Palette** onto the design workspace.
2. Right-click on the **tApacheLogInput** component and connect it to the **tLogRow** component using a **Main Row** link.
3. In the design workspace, select **tApacheLogInput**.
4. Click the **Component** tab to define the basic settings for **tApacheLogInput**.

5. If desired, click the **Edit schema** button to see the read-only columns.
6. In the **File Name** field, enter the file path or browse to the access-log file you want to read.
7. In the design workspace, select **tLogRow** and click the **Component** tab to define its basic settings. For more information, see **tLogRow** on page 1988.
8. Press **F6** to execute the Job.

```
194.04.172.41-|-[27/May/2007|03:06:12]+0200|GET|/forum/index.php|HTTP-1.0|200|19024|http://www.talendforge.org/forum/index.php|Mozilla/4.0 (compatible; MSIE 5.0; Windows NT 5.1; .NET CLR 1.1.4322)
88.154.28.65-|-[27/May/2007|05:44:21]+0200|GET|/bugs/view.php?id=205|HTTP/1.0|200|19406|http://le-gail.net/pierrick/blog/|Mozilla/4.0 (compatible; MSIE 5.0; Windows 98)
76.171.68.99-|-[27/May/2007|19:12:53]+0200|GET|/forum/login.php|HTTP/1.0|200|17445|http://talendforge.org/login.php|Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) Opera 7.54 [en]
Job apach ended at 12:11 01/09/2008. {exit code=0}
```

**Results**

The log lines of the defined file are displayed on the console.
tAS400Close

Closes the transaction committed in the connected database.

tAS400Close Standard properties

These properties are used to configure tAS400Close running in the Standard Job framework.
The Standard tAS400Close component belongs to the Databases family.
The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tAS400Connection component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with AS/400 components, especially with tAS400Connection and tAS400Commit.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [*] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-
Related scenario

No scenario is available for the Standard version of this component yet.
tAS400Commit

Commits in one go a global transaction instead of doing that on every row or every batch, and provides gain in performance, using a unique connection.
tAS400Commit validates the data processed through the Job into the connected database.

**tAS400Commit Standard properties**

These properties are used to configure tAS400Commit running in the Standard Job framework.
The Standard tAS400Commit component belongs to the Databases family.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Component list</strong></td>
<td>Select the <strong>tAS400Connection</strong> component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>
| **Close Connection** | This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.  

**Warning:**
*If you want to use a **Row > Main** connection to link **tAS400Commit** to your Job, your data will be committed row by row. In this case, do not select the **Close connection** check box or your connection will be closed before the end of your first row commit.* |

**Advanced settings**

| **tStat Catcher Statistics** | Select this check box to collect log data at the component level. |

**Usage**

<p>| <strong>Usage rule</strong> | This component is more commonly used with other tAS400* components, especially with the <strong>tAS400Connection</strong> and <strong>tAS400Rollback</strong> components. |</p>
<table>
<thead>
<tr>
<th>Dynamic settings</th>
</tr>
</thead>
</table>
| Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Related scenario

For a similar scenario using other database, see Scenario: Inserting data in mother/daughter tables on page 2476.
**tAS400Connection**

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

`tAS400Connection` opens a connection to the database for a current transaction.

**tAS400Connection Standard properties**

These properties are used to configure `tAS400Connection` running in the Standard Job framework.

The `tAS400Connection` component belongs to the Databases and the ELT families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the AS/400 version in use</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address</td>
</tr>
<tr>
<td>Database Name</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td>Use or register a shared DB Connection</td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the <strong>Shared DB Connection Name</strong> field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.</td>
</tr>
</tbody>
</table>
This option is incompatible with the Use dynamic job and Use an independent process to run subjob options of the tRunJob component. Using a shared connection together with a tRunJob component with either of these two options enabled will cause your Job to fail.

**Advanced settings**

**Additional JDBC parameters**
Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the Use an existing connection check box in the Basic settings.

**Auto Commit**
Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

**tStatCatcher Statistics**
Select this check box to gather the job processing metadata at a Job level as well as at each component level.

**Usage**

**Usage rule**
This component is more commonly used with other tAS400* components, especially with the tAS400Commit and tAS400Rollback components.

**Related scenario**

For similar scenarios using other database, see tMysqlConnection on page 2475.
tAS400Input

Reads a database and extracts fields based on a query.

tAS400Input executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Row > Main link.

tAS400Input Standard properties

These properties are used to configure tAS400Input running in the Standard Job framework.

The Standard tAS400Input component belongs to the Databases family.

The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

Note:
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

Property type

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
</tbody>
</table>
**Repository**: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see *Talend Studio User Guide*.

### DB Version
Select the AS 400 version in use

### Host
Database server IP address

### Port
Listening port number of DB server.

### Database
Name of the database

### Username and Password
DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

### Schema and Edit Schema
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

- **Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

- **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.

- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.
**Query type and Query**

Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Additional JDBC parameters</strong></th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the Use an existing connection check box in the Basic settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trim all the String/Char columns</strong></td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</td>
</tr>
<tr>
<td><strong>Trim column</strong></td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th><strong>Usage rule</strong></th>
<th>This component offers the flexibility of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
</table>
| **Dynamic settings** | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.  

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

**Scenario: Handling data with AS/400**

This scenario describes a Job that writes the user information into AS/400, and then reads the information in AS/400 and displays it on the console.
Adding and linking the components

Procedure
1. Create a new Job and add a tFixedFlowInput component, a tAS400Output component, a tAS400Input component, and a tLogRow component by typing their names in the design workspace or dropping them from the Palette.
2. Connect tFixedFlowInput to tAS400Output using a Row > Main connection.
3. Do the same to connect tAS400Input to tLogRow.
4. Connect tFixedFlowInput to tAS400Input using a Trigger > OnSubjobOk connection.

Configuring the components

Writing the data into AS/400

Procedure
1. Double-click tFixedFlowInput to open its Basic settings view.
2. Click the [...] button next to Edit schema and in the [Schema] dialog box define the schema by adding three columns: id of Integer type, and name and city of String type.

Click OK to close the [Schema] dialog box and accept the propagation prompted by the pop-up dialog box.

3. In the Mode area, select Use Inline Content (delimited file) and enter the following user information in the Content field.

1;George;Bismarck
2;Abraham;Boise
3;Taylor;Nashville
4;William;Jefferson City
5;Alexander;Jackson
6;James;Boise
7;Gerald;Little Rock
8;Tony;Richmond
9;Thomas;Springfield
10;Andre;Nashville
4. Double-click **tAS400Output** to open its **Basic settings** view.

5. In the **Host, Database, Username** and **Password** fields, enter the information required for the connection to AS/400.

6. In the **Table** field, specify the table into which you want to write the data. In this example, it is `doct1018`.

7. Select **Drop table if exists and create** from the **Action on table** drop-down list, and select **Insert** from the **Action on data** drop-down list.

**Retrieving the data from AS/400**

**Procedure**

1. Double-click **tAS400Input** to open its **Basic settings** view.

2. In the **Host, Database, Username** and **Password** fields, enter the information required for the connection to AS/400.
3. Click the [...] button next to Edit schema and in the [Schema] dialog box define the schema by adding three columns: id of Integer type, and name and city of String type. The data structure is same as the structure you have defined for tFixedFlowInput.

4. In the Table Name field, enter or browse to the table into which you write the data. In this example, it is doct1018.

5. In the Query field, enter the SQL query sentence to be used to retrieve the user data from AS/400. In this example, it is SELECT * FROM doct1018.

6. Double-click tLogRow to open its Basic settings view.

7. In the Mode area, select Table (print values in cells of a table) for better readability of the result.

**Saving and executing the Job**

**Procedure**

1. Press Ctrl + S to save the Job.

2. Press F6 or click Run on the Run tab to run the Job.

   ```
   [statistics] connecting to socket on port 3565
   [statistics] connected
   +----------+
   | tLogRow_1 |
   +----------+
   | id   | name      | city    |
   +----------+
   | 1    | George    | Bismarck|
   | 2    | Abraham   | Boise   |
   | 3    | Taylor    | Nashville|
   | 4    | William   | Jefferson City|
   | 5    | Alexander | Jackson |
   | 6    | James     | Boise   |
   | 7    | Gerald    | Little Rock|
   | 8    | Tony      | Richmond |
   | 9    | Thomas    | Springfield|
   | 10   | Andre     | Nashville|
   +----------+
   [statistics] disconnected
   ```

   As shown above, the user information is written into AS/400, and then the data is retrieved from AS/400 and displayed on the console.

**Related scenarios**

For similar scenarios using other databases, see:
Related topic in **tContextLoad**, see Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
tAS400LastInsertId

Obtains the primary key value of the record that was last inserted in an AS/400 table.
tAS400LastInsertId fetches the last inserted ID from a selected AS/400 Connection.

**tAS400LastInsertId Standard properties**

These properties are used to configure tAS400LastInsertId running in the Standard Job framework.
The Standard tAS400LastInsertId component belongs to the Databases family.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema</strong> and <strong>Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and job flow charts. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
<td></td>
</tr>
<tr>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the <strong>[Repository Content]</strong> window.</td>
<td></td>
</tr>
</tbody>
</table>
Component list

Select the relevant **tAS400Connection** component in the list if more than one connection is planned for the current job.

Advanced settings

**tStat Catcher Statistics**

Select this check box to collect log data at the component level.

Usage

**Usage rule**

This component is to be used as an intermediary component.

**Dynamic settings**

Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

Related scenario

For a similar scenario using other database, see Scenario: Get the ID for the last inserted record with **tMysqlLastInsertId** on page 2510.
tAS400Output

Writes, updates, makes changes or suppresses entries in a database.
tAS400Output executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

**tAS400Output Standard properties**

These properties are used to configure tAS400Output running in the Standard Job framework.

The Standard tAS400Output component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><img src="icon.png" alt="Icon" /></td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component <strong>Basic settings</strong> view.</td>
</tr>
<tr>
<td></td>
<td>For more information about setting up and storing database connection parameters, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td><strong>DB Version</strong></td>
<td>Select the AS/400 version in use</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic**
settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](#).

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
</tbody>
</table>
| Action on table | On the table defined, you can perform one of the following operations:  
None: No operation is carried out.  
Drop and create a table: The table is removed and created again.  
Create a table: The table does not exist and gets created.  
Create a table if not exists: The table is created if it does not exist.  
Drop a table if exists and create: The table is removed if it already exists and created again.  
Clear a table: The table content is deleted. |
| Action on data |  |
| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
Built-In: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](#).  
Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](#).  
When the schema to be reused has default values that are integers or functions, ensure that these default values are correct. |

values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Die on error**

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

**Advanced settings**

**Use commit control**

Select this check box to have access to the **Commit every** field where you can define the commit operation.

**Commit every**: Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.

**Additional JDBC parameters**

Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the Basic settings.

**Note:**

You can press **Ctrl+Space** to access a list of predefined global variables.

**Additional Columns**

This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.

**Name**: Type in the name of the schema column to be altered or inserted as new column
**SQL expression**: Type in the SQL statement to be executed in order to alter or insert the relevant column data.

**Position**: Select **Before**, **Replace** or **After** following the action to be performed on the reference column.

**Reference column**: Type in a column of reference that the tDBOutput can use to place or replace the new or altered column.

**Use field options**
Select this check box to customize a request, especially when there is double action on data.

**Enable debug mode**
Select this check box to display each step during processing entries in a database.

**Use Batch**
Select this check box to activate the batch mode for data processing.

**Note:**
This check box is available only when you have selected the **Insert**, **Update** or **Delete** option in the **Action on data** field.

**Batch Size**
Specify the number of records to be processed in each batch.
This field appears only when the **Use batch mode** check box is selected.

**tStat Catcher Statistics**
Select this check box to collect log data at the component level.

**Usage**

**Usage rule**
This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.
This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in an AS/400 database. It also allows you to create a reject flow using a **Row > Reject** link to filter data in error. For an example of tMySqlOutput in use, see **Scenario: Retrieving data in error with a Reject link** on page 2529.

**Dynamic settings**
Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.
The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Related scenarios**

For related scenario, see **Scenario: Handling data with AS/400** on page 233.

For similar scenarios using other databases, see:

- **Scenario: Inserting a column and altering data using tMysqlOutput** on page 2521.
tAS400Rollback

Cancels the transaction commit in the connected database and avoids to commit part of a transaction involuntarily.

**tAS400Rollback Standard properties**

These properties are used to configure tAS400Rollback running in the Standard Job framework. The Standard tAS400Rollback component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tAS400Connection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tAS400* components, especially with the tAS400Connection and tAS400Commit components. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |
Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see [Scenario: Reading data from databases through context-based dynamic connections](#) on page 2497 and [Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters](#) on page 489. For more information on **Dynamic settings** and context variables, see [Talend Studio User Guide](#).

---

**Related scenarios**

For a similar scenario using other database, see [Scenario: Rollback from inserting data in mother/daughter tables](#) on page 2480.
tAS400Row

Executes the SQL query stated onto the specified database.

Depending on the nature of the query and the database, tAS400Row acts on the actual DB structure or on the data (although without handling data). The SQLBuilder tool helps you write easily your SQL statements. tAS400Row is the specific component for this database query. The row suffix means the component implements a flow in the job design although it does not provide output.

**tAS400Row Standard properties**

These properties are used to configure tAS400Row running in the Standard Job framework.

The Standard tAS400Row component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components on page 598](#).

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.
| **DB Version** | Select the AS/400 version in use |
| **Host** | Database server IP address |
| **Port** | Listening port number of DB server. |
| **Database** | Name of the database |
| **Username and Password** | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields. |

**Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

| **Query type** | Either **Built-in** or **Repository**. |

**Built-in**: Fill in manually the query statement or build it graphically using SQLBuilder

**Repository**: Select the relevant query stored in the Repository. The Query field gets accordingly filled in.
**Query**

Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

**Die on error**

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

---

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Additional JDBC Parameters</strong></th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <strong>Use an existing connection</strong> check box in the <strong>Basic settings</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Propagate QUERY's recordset</strong></td>
<td>Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the <strong>use column</strong> list.</td>
</tr>
</tbody>
</table>

**Note:**

This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY's recordset should be set to the type of **Object** and this component is usually followed by **tParseRecordSet**.

<table>
<thead>
<tr>
<th><strong>Use PreparedStatement</strong></th>
<th>Select this check box if you want to query the database using a PreparedStatement. In the <strong>Set PreparedStatement Parameter</strong> table, define the parameters represented by '?' in the SQL instruction of the <strong>Query</strong> field in the <strong>Basic Settings</strong> tab.</th>
</tr>
</thead>
</table>

**Parameter Index:** Enter the parameter position in the SQL instruction.

**Parameter Type:** Enter the parameter type.

**Parameter Value:** Enter the parameter value.

**Note:**

This option is very useful if you need to execute the same query several times. Performance levels are increased.

<table>
<thead>
<tr>
<th><strong>Commit every</strong></th>
<th>Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.</th>
</tr>
</thead>
</table>

| **tStat Catcher Statistics** | Select this check box to collect log data at the component level.                                                                                                                                   |
### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
</table>

**Dynamic settings**

Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Related scenarios

For similar scenarios using other databases, see:

- **Scenario: Combining two flows for selective output** on page 2558.
- **Procedure** on page 625.
- **Scenario: Removing and regenerating a MySQL table index** on page 2552.
tAssert

Generates the boolean evaluation on the concern for the Job execution status and provides the Job status messages to tAssertCatcher.

The status includes:

• **Ok**: the Job execution succeeds.
• **Fail**: the Job execution fails.

The tested Job’s result does not match the expectation or an execution error occurred at runtime.

The tAssert component works alongside tAssertCatcher to evaluate the status of a Job execution. It concludes with the boolean result based on an assertive statement related to the execution and feed the result to tAssertCatcher for proper Job status presentation.

**tAssert Standard properties**

These properties are used to configure tAssert running in the Standard Job framework.

The Standard tAssert component belongs to the Logs & Errors family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Description</th>
<th>Type in your descriptive message to help identify the assertion of a tAssert.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>Type in the assertive statement you base the evaluation on.</td>
</tr>
</tbody>
</table>

**Global Variables**

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

**Usage**

| Usage rule | This component follows the action the assertive condition is directly related to. It can be the |
### Scenario 1: Viewing product orders status (on a daily basis) against a benchmark number

This scenario allows you to insert the orders information into a database table and to evaluate the orders status (every day once scheduled to run) by using `tAssert` to compare the orders against a fixed number and `tAssertCatcher` to indicate the results. In this case, `Ok` is returned if the number of orders is greater than 20 and `Failed` is returned if the number of orders is less than 20.

In practice, this Job can be scheduled to run every day for the daily orders report and `tFixedFlowInput` as well as `tLogRow` are replaced by input and output components in the `Database/File` families.

### Linking the components

**Procedure**

1. Drop `tFixedFlowInput`, `tMysqlOutput`, `tAssert`, `tAssertCatcher`, and `tLogRow` onto the workspace.
2. Rename `tFixedFlowInput` as `orders`, `tAssert` as `orders >=20`, `tAssertCatcher` as `catch comparison result` and `tLogRow` as `ok or failed`.
3. Link `tFixedFlowInput` to `tMysqlOutput` using a `Row > Main` connection.
4. Link `tFixedFlowInput` to `tAssert` using the `Trigger > On Subjob OK` connection.
5. Link `tAssertCatcher` to `tLogRow` using a `Row > Main` connection.

### Configuring the components

**Procedure**

1. Double-click `tFixedFlowInput` to open its `Basic settings` view.
Select **Use Inline Content (delimited file)** in the **Mode** area.

In the **Content** field, enter the data to write to the Mysql database, for example:

```
AS2152;Washington Berry Juice;2013-02-19 11:14:15;3.6
AS2152;Washington Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washington Berry Juice;2013-02-19 13:14:15;3.6
AS2152;Washington Berry Juice;2013-02-19 14:14:15;3.6
AS2152;Washington Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washington Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washington Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washington Berry Juice;2013-02-19 12:14:15;3.6
AS2152;Washington Berry Juice;2013-02-19 12:14:15;3.6
```

Note that the orders listed are just for illustration of how **tAssert** functions and the number here is less than 20.

2. Click the **Edit schema** button to open the schema editor.
3. Click the [+] button to add four columns, namely `product_id`, `product_name`, `date` and `price`, of the `String`, `Date`, `Float` types respectively.
   Click **OK** to validate the setup and close the editor.

4. Double-click **tMysqlOutput** to display the **Basic settings** view.

5. In the **Host**, **Port**, **Database**, **Username** and **Password** fields, enter the connection details and the authentication credentials.

6. In the **Table** field, enter the name of the table, for example **order**.

7. In the **Action on table** list, select the option **Drop table if exists and create**.

8. In the **Action on data** list, select the option **Insert**.

9. Double-click **tAssert** to display the **Basic settings** view.
10. In the **description** field, enter the descriptive information for the purpose of **tAssert** in this case.

11. In the **expression** field, enter the expression allowing you to compare the data to a fixed number:

```java
((Integer)globalMap.get("tMysqlOutput_1_NB_LINE_INSERTED")) >= 20
```

12. Double-click **tLogRow** to display the **Basic settings** view.

13. In the **Mode** area, select **Table (print values in cells of a table)** for a better display.

**Executing the Job**

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to run the Job.

As shown above, the orders status indicates *Failed* as the number of orders is less than 20.

**Scenario 2: Setting up the assertive condition for a Job execution**

This scenario describes how to set up an assertive condition in **tAssert** in order to evaluate that a Job execution succeeds or not. Moreover, you can also find out how the two different evaluation results display and the way to read them. Apart from **tAssert**, the scenario uses the following components as well:

- **tFileInputDelimited** and **tFileOutputDelimited**. The two components compose the main Job of which the execution status is evaluated. For the detailed information on the two components, see **tFileInputDelimited** on page 1010 and **tFileOutputDelimited** on page 1110.
• **tFileCompare.** It realizes the comparison between the output file of the main Job and a standard reference file. The comparative result is evaluated by **tAssert** against the assertive condition set up in its settings. For more detailed information on **tFileCompare**, see **tFileCompare** on page 980.

• **tAssertCatcher.** It captures the evaluation generated by **tAssert.** For more information on **tAssertCatcher**, see **tAssertCatcher** on page 263.

• **tLogRow.** It allows you to read the captured evaluation. For more information on **tLogRow**, see **tLogRow** on page 1988.

First proceed as follows to design the main Job:

• Prepare a delimited .csv file as the source file read by your main Job.

• Edit two rows in the delimited file. The contents you edit are not important, so feel free to simplify them.

• Name it **source.csv**.

• In **Talend Studio**, create a new job **JobAssertion**.

• Place **tFileInputDelimited** and **tFileOutputDelimited** on the workspace.

• Connect them with a **Row Main** link to create the main Job.

• Double-click **tFileInputDelimited** to open its **Component** view.

• In the **File Name** field of the **Component** view, fill in the path or browse to **source.csv**.

• Still in the **Component** view, set **Property Type** to **Built-In** and click **...** next to **Edit schema** to define the data to pass on to **tFileOutputDelimited**. In the scenario, define the data presented in **source.csv** you created.

For more information about schema types, see **Talend Studio User Guide**.

• Define the other parameters in the corresponding fields according to **source.csv** you created.

• Double-click **tFileOutputDelimited** to open its **Component** view.
• In the **File Name** field of the **Component** view, fill in or browse to specify the path to the output file, leaving the other fields as they are by default.

• Press **F6** to execute the main Job. It reads `source.csv`, pass the data to **tFileOutputDelimited** and output an delimited file, `out.csv`.

Then continue to edit the Job to see how **tAssert** evaluates the execution status of the main Job.

• Rename `out.csv` as `reference.csv`. This file is used as the expected result the main Job should output.

• Place **tFileCompare**, **tAssert** and **tLogRow** on the workspace.

• Connect them with **Row Main** link.

• Connect **tFileInputDelimited** to **tFileCompare** with **OnSubjobOk** link.

• Double-click **tFileCompare** to open its **Component** view.

• In the **Component** view, fill in the corresponding file paths in the **File to compare** field and the **Reference file** field, leaving the other fields as default.
For more information on the tFileCompare component, see tFileCompare on page 980.

- Then click tAssert and click the Component tab on the lower side of the workspace.

- In the Component view, edit the assertion `row2.differ==0` in the expression field and the descriptive message of the assertion in description field.

In the expression field, `row2` is the data flow transmitting from tFileCompare to tAssert, differ is one of the columns of the tFileCompare schema and presents whether the compared files are identical, and `0` means no difference is detected between the `out.csv` and `reference.csv` by tFileCompare. Hence when the compared files are identical, the assertive condition is thus fulfilled, tAssert concludes that the main Job succeeds; otherwise, it concludes failure.

⚠️ Note:

The differ column is in the read-only tFileCompare schema. For more information on its schema, see tFileCompare on page 980.

- Press F6 to execute the Job.
- Check the result presented in the Run view
The console shows the comparison result of tFileCompare: Files are identical. But you find nowhere the evaluation result of tAssert.

So you need tAssertCatcher to capture the evaluation.

- Place tAssertCatcher and tLogRow on the workspace.
- Connect them with Row Main link.

- Use the default configuration in the Component view of tAssertCatcher.
• Press **F6** to execute the Job.

• Check the result presented in the **Run** view. You will see the Job status information is added in:

```
2010-01-29 15:37:33|fAvAzH|TASSERT|JobAssertion|java|tAssert_1|Ok|--|
The output file should be identical with the reference file
```

The descriptive information on **JobAssertion** in the console is organized according to the **tAssertCatcher** schema. This schema includes, in the following order, the execution time, the process ID, the project name, the Job name, the code language, the evaluation origin, the evaluation result, detailed information of the evaluation, descriptive message of the assertion. For more information on the schema of **tAssertCatcher**, see [tAssertCatcher on page 263](#).

The console indicates that the execution status of **JobAssertion** is **Ok**. In addition to the evaluation, you can still see other descriptive information about **JobAssertion** including the descriptive message you have edited in the **Basic settings** of **tAssert**.

Then you will perform operations to make the main Job fail to generate the expected file. To do so, proceed as follows in the same Job you have executed:

• Delete a row in **reference.csv**.

• Press **F6** to execute the Job again.

• Check the result presented in the **Run** view.

```
2010-02-01 19:47:43|GeHJNO|TASSERT|JobAssertion|tAssert_1|Failed|Test logically failed|The output file should be identical with the reference file
```

The console shows that the execution status of the main Job is **Failed**. The detailed explanation for this status is closely behind it, reading **Test logically failed**.

You can thus get a basic idea about your present Job status: it fails to generate the expected file because of a logical failure. This logical failure could come from a logical mistake during the Job design.
The status and its explanatory information are presented respectively in the *status* and the *substatus* columns of the `tAssertCatcher` schema. For more information on the columns, see `tAssertCatcher` on page 263.
tAssertCatcher

Generates a data flow consolidating the status information of a job execution and transfer the data into defined output files.

Based on its pre-defined schema, tAssertCatcher fetches the execution status information from repository, Job execution and tAssert.

**tAssertCatcher Standard properties**

These properties are used to configure tAssertCatcher running in the Standard Job framework.

The Standard tAssertCatcher component belongs to the Logs & Errors family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description, it defines the fields to be processed and passed on to the next component. In this particular case, the schema is read-only, as this component gathers standard log information including:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Moment:</strong> Processing time and date.</td>
</tr>
<tr>
<td></td>
<td><strong>Pid:</strong> Process ID.</td>
</tr>
<tr>
<td></td>
<td><strong>Project:</strong> Project which the job belongs to.</td>
</tr>
<tr>
<td></td>
<td><strong>Job:</strong> Job name.</td>
</tr>
<tr>
<td></td>
<td><strong>Language:</strong> Language used by the Job (Java)</td>
</tr>
<tr>
<td></td>
<td><strong>Origin:</strong> Status evaluation origin. The origin may be different tAssert components.</td>
</tr>
<tr>
<td></td>
<td><strong>Status:</strong> Evaluation fetched from tAssert. They may be</td>
</tr>
<tr>
<td></td>
<td>- <strong>Ok:</strong> if the assertive statement of tAssert is evaluated as true at runtime.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Failed:</strong> if the assertive statement of tAssert is evaluated as false or an execution error occurs at runtime. The tested Job's result does not match the expectation or an execution error occurred at runtime.</td>
</tr>
<tr>
<td></td>
<td><strong>Substatus:</strong> Detailed explanation for failed execution. The explanation can be:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Test logically failed:</strong> the investigated Job does not produce the expected result.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Execution error:</strong> an execution error occurred at runtime.</td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong> Descriptive message typed in Basic settings of tAssert (when Catch tAssert is selected) and/or the</td>
</tr>
</tbody>
</table>
message of the exception captured (when **Catch Java Exception** is selected).

**Exception**: The Exception object thrown by the Job, namely the original exception.
Available when **Get original exception** is selected.

**Catch Java Exception**
- This check box allows to capture Java exception errors and show the message in the **Description** column (**Get original exception** not selected) or in the **Exception** column (**Get original exception** selected) column, once checked.

**Get original exception**
- This check box allows to show the original exception object in the **Exception** column, once checked.
Available when **Catch Java Exception** is selected.

**Catch tAssert**
- This check box allows to capture the evaluations of **tAssert**.

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.

**Usage**

**Usage rule**
- This component is the start component of a secondary Job which fetches the execution status information from several sources. It generates a data flow to transfer the information to the component which proceeds.

**Limitation**
- This component must be used with **tAssert** together.

**Related scenarios**

For using case in relation with **tAssertCatcher**, see **tAssert** scenario:

- **Scenario 2: Setting up the assertive condition for a Job execution** on page 256
tAzureStorageConnection

Uses authentication and the protocol information to create a connection to the Microsoft Azure Storage system that can then be reused by other Azure Storage components.

### tAzureStorageConnection Standard properties

These properties are used to configure tAzureStorageConnection running in the Standard Job framework.

The Standard tAzureStorageConnection component belongs to the Cloud family.

The component in this framework is generally available.

#### Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Account Name</th>
<th>Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Account Key</th>
<th>Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Select the protocol for this connection to be created.</th>
</tr>
</thead>
</table>

| Use Azure Shared Access Signature | Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS). In the Azure Shared Access Signature field displayed, enter your account SAS URL between double quotation marks. You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is https://<$storagename>$<service>.core.windows.net/<$sastoken>, where <$storagename> is the storage account name, <$service> is the allowed service name (blob,
AzureStorageConnection

file, queue or table), and <sastoken> is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

Global Variables

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

Usage

| Usage rule | This component is generally used with other Azure Storage components.
Knowledge about Microsoft Azure Storage is required. |

Related scenario

For related scenarios, see:

- Scenario: Retrieving files from a Azure Storage container on page 285
- Scenario: Creating a container in Azure Storage on page 269
- Handling data with Microsoft Azure Table storage on page 295
tAzureStorageContainerCreate

Creates a new storage container used to hold Azure blobs (Binary Large Object) for a given Azure storage account.

**tAzureStorageContainerCreate Standard properties**

These properties are used to configure tAzureStorageContainerCreate running in the Standard Job framework.

The Standard tAzureStorageContainerCreate component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the <strong>Connection Component</strong> drop-down list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list.</th>
</tr>
</thead>
</table>

| Account Name | Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account. |

| Account Key | Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access. |

| Protocol | Select the protocol for this connection to be created. |

| Use Azure Shared Access Signature | Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS). |
In the **Azure Shared Access Signature** field displayed, enter your account SAS URL between double quotation marks. You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is `https://<$storagename>.$service>.core.windows.net/<$sastoken>`, where `<$storagename>` is the storage account name, `<$service>` is the allowed service name (blob, file, queue or table), and `<$sastoken>` is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

<table>
<thead>
<tr>
<th>Container name</th>
<th>Enter the name of the blob container you need to create.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access control</td>
<td>Select the access restriction level you need to apply on the container to be created.</td>
</tr>
<tr>
<td>Die on error</td>
<td>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row &gt; Reject link.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| CONTAINER | The name of the blob container. This is an After variable and it returns a string. |
| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

**Usage**

| Usage rule | This component can be used as a standalone component of a Job or Subjob. |
| Prerequisites | Knowledge about Microsoft Azure Storage is required. |
Scenario: Creating a container in Azure Storage

In this scenario, a four-component Job uses Azure Storage components to create a container in a given Azure Storage system and check whether this container is successfully created.

Before replicating this scenario, you must have appropriate rights and permissions to read and write files in the Azure storage account to be used. For further information, see Microsoft’s documentation for Azure Storage: http://azure.microsoft.com/en-us/documentation/services/storage/.

Linking the components

Procedure
1. In the Integration perspective of the Studio, create an empty Job, named azureTalend for example, from the Job Designs node in the Repository tree view.
2. Drop tAzureStorageConnection, tAzureStorageContainerCreate, tAzureStorageContainerExist and tJava onto the workspace.
3. Connect them using the Trigger > OnSubjobOk link.

Connecting to an Azure storage account

Procedure
1. Double-click tAzureStorageConnection to open its Component view.
2. In the Account name field, enter the name of the storage account to be connected to. In this example, it is talendstorage, an account that has been created for demonstration purposes.
3. In the Account key field, paste the primary or the secondary key associated with the storage account to be used. These keys can be found in the Manage Access Key dashboard in the Azure Storage system to be connected to.
4. From the Protocol list, select the protocol for the endpoint of the storage account to be used. In this example, it is HTTPS.

Creating a container

Procedure

1. Double-click tAzureStorageContainerCreate to open its Component view.

2. Select the component whose connection details will be used to set up the Azure storage connection. In this example, it is tAzureStorageConnection_1.

3. In the Container name field, enter the name of the container you need to create. If a container using the same name exists, that container will be overwritten at runtime.

4. From the Access control list, select the access restriction level for the container to be created. In this example, select Private.

Verifying the creation

Procedure

1. Double-click tAzureStorageContainerExist to open its Component view.

2. Select the component whose connection details will be used to set up the Azure storage connection. In this example, it is tAzureStorageConnection_1.

3. In the Container name field, enter the name of the container you need to check whether it exists.

4. Double-click tJava to open its Component view.

5. In the Code field, enter System.out.println();
6. In the **Outline** panel, which, by default, is found to the left side of the **Component** view, expand the `tAzureStorageContainerExist` node.

![Outline Panel with tAzureStorageContainerCreate node](image)

7. From the **Outline** panel, drop the `CONTAINER_EXIST` global variable into the parentheses in the code in the **Component** view in order to make the code read:

   ```java
   System.out.println(((Boolean) globalMap.get("tAzureStorageContainerExist\_1\_CONTAINER\_EXIST"));
   ```

### Executing the Job

**Procedure**

1. Press **F6** to run this Job.
2. Check the execution result on the **Run** console.

![Execution Console](image)

   Starting job azureTalend at 17.49
   06/05/2014
   [statistics] connecting to socket on port 3630
   [statistics] connected
   true
   [statistics] disconnected
   Job azureTalend ended at 17.49
   06/05/2014. [exit code=0]

   You can read that the Job returns **true** as the verification result, that is to say, the `talendcontainer` container has been created in the storage account being used.

3. Double-check the result in the web console of the Azure storage account.
You can read as well that the talendcontainer container has been created.
**tAzureStorageContainerDelete**

Automates the removal of a given blob container from the space of a specific storage account.

**tAzureStorageContainerDelete Standard properties**

These properties are used to configure tAzureStorageContainerDelete running in the Standard Job framework.

The Standard tAzureStorageContainerDelete component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

This property is not available when other connection component is selected from the Connection Component drop-down list.

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account Name</td>
<td>Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account.</td>
</tr>
<tr>
<td>Account Key</td>
<td>Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Select the protocol for this connection to be created.</td>
</tr>
<tr>
<td>Use Azure Shared Access Signature</td>
<td>Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS).</td>
</tr>
</tbody>
</table>

In the **Azure Shared Access Signature** field displayed, enter your account SAS URL between double quotation marks.
You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is https://<$storagename>..<$service>.core.windows.net/<$sastoken>, where <$storagename> is the storage account name, <$service> is the allowed service name (blob, file, queue or table), and <$sastoken> is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

**Container name**
Enter the name of the blob container to be removed.

**Die on error**
Select the check box to stop the execution of the Job when an error occurs.
Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row > Reject link.

**Advanced settings**

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

**CONTAINER**
The name of the blob container. This is an After variable and it returns a string.

**ERROR_MESSAGE**
The error message generated by the component when an error occurs. This is an After variable and it returns a string.

**Usage**

**Usage rule**
This component can be used as a standalone component of a Job or Subjob.

**Prerequisites**
Knowledge about Microsoft Azure Storage is required.

**Related scenarios**

No scenario is available for the Standard version of this component yet.
**tAzureStorageContainerExist**

Automates the verification of whether a given blob container exists or not within a storage account.

**tAzureStorageContainerExist Standard properties**

These properties are used to configure tAzureStorageContainerExist running in the Standard Job framework.

The Standard tAzureStorageContainerExist component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in. This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account Name</td>
<td>Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account.</td>
</tr>
<tr>
<td>Account Key</td>
<td>Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Select the protocol for this connection to be created.</td>
</tr>
<tr>
<td>Use Azure Shared Access Signature</td>
<td>Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS). In the Azure Shared Access Signature field displayed, enter your account SAS URL between double quotation marks.</td>
</tr>
</tbody>
</table>
marks. You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is https://<storagename>.<service>.core.windows.net/<sastoken>, where <storagename> is the storage account name, <service> is the allowed service name (blob, file, queue or table), and <sastoken> is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

<table>
<thead>
<tr>
<th>Container name</th>
<th>Enter the name of the blob container you need to verify whether it exists.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die on error</td>
<td>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row &gt; Reject link.</td>
</tr>
</tbody>
</table>

**Advanced settings**

- **tStatCatcher Statistics**: Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

- **CONTAINER**: The name of the blob container. This is an After variable and it returns a string.
- **CONTAINER_EXIST**: The result of whether the given container exists or not. This is an After variable and it returns a boolean.
- **ERROR_MESSAGE**: The error message generated by the component when an error occurs. This is an After variable and it returns a string.

**Usage**

- **Usage rule**: This component can be used as a standalone component of a Job or Subjob.
- **Prerequisites**: Knowledge about Microsoft Azure Storage is required.

**Related scenario**

For a related scenario, see Scenario: Creating a container in Azure Storage on page 269
**tAzureStorageContainerList**

Lists all containers in a given Azure storage account.

**tAzureStorageContainerList Standard properties**

These properties are used to configure tAzureStorageContainerList running in the Standard Job framework.

The Standard tAzureStorageContainerList component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Account Name</th>
<th>Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Account Key</th>
<th>Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Select the protocol for this connection to be created.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Use Azure Shared Access Signature</th>
<th>Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In the <strong>Azure Shared Access Signature</strong> field displayed, enter your account SAS URL between double quotation</td>
</tr>
</tbody>
</table>
marks. You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is https://<storagename>.<service>.core.windows.net/<sastoken>, where <storagename> is the storage account name, <service> is the allowed service name (blob, file, queue or table), and <sastoken> is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

The schema of this component is predefined with a single column **ContainerName** of String type, which indicates the name of each container to be listed.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.

- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Die on error

Select the check box to stop the execution of the Job when an error occurs.

### Advanced settings

- **tStatCatcher Statistics**: Select this check box to gather the Job processing metadata at the Job level as well as at each component level.
Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong></td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage rule</strong></td>
<td>This component is usually used as a start component of a Job or Subjob and it always needs an output link.</td>
</tr>
<tr>
<td><strong>Prerequisites</strong></td>
<td>Knowledge about Microsoft Azure Storage is required.</td>
</tr>
</tbody>
</table>

Related scenario

No scenario is available for this component yet.
tAzureStorageDelete

Deletes blobs from a given container for an Azure storage account according to the specified blob filters.

tAzureStorageDelete Standard properties

These properties are used to configure tAzureStorageDelete running in the Standard Job framework. The Standard tAzureStorageDelete component belongs to the Cloud family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
</tbody>
</table>

| Connection Component | Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list. |

| Account Name | Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account. |

| Account Key | Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access. |

| Protocol | Select the protocol for this connection to be created. |

| Use Azure Shared Access Signature | Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS). In the Azure Shared Access Signature field displayed, enter your account SAS URL between double quotation marks. |
You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is https://<$storagename>.$service.core.windows.net/<$sastoken>, where <$storagename> is the storage account name, <$service> is the allowed service name (blob, file, queue or table), and <$sastoken> is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

<table>
<thead>
<tr>
<th>Container name</th>
<th>Enter the name of the container from which you need to delete blobs.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Blob filter</strong></td>
<td>Complete this table to select the blobs to be deleted. The parameters to be provided are:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Blob prefix</strong>: enter the common prefix of the names of the blobs you need to delete. This prefix allows you to filter the blobs which have the specified prefix in their names in the given container.</td>
</tr>
<tr>
<td></td>
<td>A blob name contains the virtual hierarchy of the blob itself. This hierarchy is a virtual path to that blob and is relative to the container where that blob is stored. For example, in a container named photos, the name of a photo blob might be 2014/US/Oakland/Talend.jpg.</td>
</tr>
<tr>
<td></td>
<td>For this reason, when you define a prefix, you are actually designating a directory level as the blob filter, for example, 2014/ or 2014/US/.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Include subdirectories</strong>: select this check box to select all of the sub-folders and the blobs in those folders beneath the designated directory level. If you leave this check box clear, tAzureStorageDelete deletes only the blobs directly beneath that directory level.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>Select the check box to stop the execution of the Job when an error occurs.</td>
</tr>
<tr>
<td></td>
<td>Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row &gt; Reject link.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |
Global Variables

<table>
<thead>
<tr>
<th>CONTAINER</th>
<th>The name of the blob container. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as a standalone component of a Job or Subjob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prerequisites</td>
<td>Knowledge about Microsoft Azure Storage is required.</td>
</tr>
</tbody>
</table>

Related scenarios

No scenario is available for the Standard version of this component yet.
tAzureStorageGet

Retrieves blobs from a given container for an Azure storage account according to the specified filters applied on the virtual hierarchy of the blobs and then write selected blobs in a local folder.

**tAzureStorageGet Standard properties**

These properties are used to configure tAzureStorageGet running in the Standard Job framework. The Standard tAzureStorageGet component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in. This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
</tbody>
</table>

| Connection Component | Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list. |

| Account Name | Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account. |

| Account Key | Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access. |

| Protocol | Select the protocol for this connection to be created. |

| Use Azure Shared Access Signature | Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS). |

In the **Azure Shared Access Signature** field displayed, enter your account SAS URL between double quotation marks.
You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is https://<storagename>.<service>.core.windows.net/<sastoken>, where <storagename> is the storage account name, <service> is the allowed service name (blob, file, queue or table), and <sastoken> is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

<table>
<thead>
<tr>
<th>Container</th>
<th>Enter the name of the container you need to retrieve blobs from.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local folder</td>
<td>Enter the path, or browse to the folder in which you need to store the retrieved blobs.</td>
</tr>
<tr>
<td>Blobs</td>
<td>Complete this table to select the blobs to be retrieved. The parameters to be provided are:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Prefix</strong>: enter the common prefix of the names of the blobs you need to retrieve. This prefix allows you to filter the blobs which have the specified prefix in their names in the given container. A blob name contains the virtual hierarchy of the blob itself. This hierarchy is a virtual path to that blob and is relative to the container where that blob is stored. For example, in a container named photos, the name of a photo blob might be <code>2014/US/Oakland/Talend.jpg</code>. For this reason, when you define a prefix, you are actually designating a directory level as the blob filter, for example, <code>2014/</code> or <code>2014/US/</code>. If you want to select the blobs stored directly beneath the container level, that is to say, the blobs without virtual path in their names, remove quotation marks and enter <code>null</code>.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Include sub-directories</strong>: select this check box to retrieve all of the sub-folders and the blobs in those folders beneath the designated directory level in the Blob prefix column. If you leave this check box clear, <code>tAzureStorageGet</code> returns only the blobs directly beneath that directory level.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Create parent directories</strong>: select this check box to replicate the virtual directory of the retrieved blobs in the local folder. Note that if you leave this check box clear, there must be the same directory in the local folder as the retrieved blobs have in the container; otherwise, those blobs cannot be retrieved.</td>
</tr>
</tbody>
</table>
Die on error

Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row > Reject link.

Advanced settings

tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Global Variables

| CONTAINER | The name of the blob container. This is an After variable and it returns a string. |
| LOCAL_FOLDER | The local directory used in this component. This is an After variable and it returns a string. |
| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

Usage

| Usage rule | This component can be used as a standalone component of a Job or Subjob. |
| Prerequisites | Knowledge about Microsoft Azure Storage is required. |

Scenario: Retrieving files from a Azure Storage container

In this scenario, a five-component Job uses Azure Storage components to write files in a given Azure Storage system and then retrieve selected files (blobs in terms of Azure Storage) from that system.
Before replicating this scenario, you must have appropriate rights and permissions to read and write files in the Azure storage account to be used. For further information, see Microsoft’s documentation for Azure Storage: http://azure.microsoft.com/en-us/documentation/services/storage/.

The `talendcontainer` container used in this scenario was created using `tAzureStorageContainerCreate` in the scenario Scenario: Creating a container in Azure Storage on page 269.

**Linking the components**

**Procedure**

1. In the **Integration** perspective of the Studio, create an empty Job, named `azureTalend` for example, from the **Job Designs** node in the **Repository** tree view.
2. Drop `tAzureStoragePut`, `tAzureStorageList`, `tJava` and `tAzureStorageGet` onto the workspace.
3. Connect the Azure Storage components using the **Trigger > OnSubjobOk** link while connect `tAzureStorageList` to `tJava` using the **Row > Iterate** link.

**Connecting to an Azure storage account**

**Procedure**

1. Double-click `tAzureStorageConnection` to open its **Component** view.
2. In the **Account name** field, enter the name of the storage account to be connected to. In this example, it is **talendstorage**, an account that has been created for demonstration purposes.

3. In the **Account key** field, paste the primary or the secondary key associated with the storage account to be used. These keys can be found in the Manage Access Key dashboard in the Azure Storage system to be connected to.

4. From the **Protocol** list, select the protocol for the endpoint of the storage account to be used. In this example, it is **HTTPS**.

**Writing files in Azure Storage**

**Procedure**

1. Double-click **tAzureStoragePut** to open its **Component** view.

2. Select the component whose connection details will be used to set up the Azure storage connection. In this example, it is **tAzureStorageConnection_1**.

3. In the **Container name** field, enter the name of the container you need to write files in. In this example, it is **talendcontainer**, a container created in the scenario Scenario: Creating a container in Azure Storage on page 269.

4. In the **Local folder** field, enter the path, or browse, to the directory where the files to be used are stored. In this scenario, they are some pictures showing technical process and stored locally in **E:/photos**. Therefore, put **E:/photos**; this allows **tAzureStoragePut** to upload all the files of this folder and its sub-folders into the **talendcontainer** container.

For demonstration purposes, the example photos are organized as follows in the **E:/photos** folder:

- **Directly beneath the E:/photos level:**
  
  - `components-use_case_triakinput_1.png`
  - `components-use_case_triakinput_2.png`
  - `components-use_case_triakinput_3.png`
  - `components-use_case_triakinput_4.png`

- **In the E:/photos/mongodb/step1 directory:**
  
  - `components-use_case_tmongodbbulkload_1.png`
  - `components-use_case_tmongodbbulkload_2.png`
  - `components-use_case_tmongodbbulkload_3.png`
  - `components-use_case_tmongodbbulkload_4.png`

- **In the E:/photos/mongodb/step2 directory:**


5. In the Azure Storage folder field, enter the directory where you want to write files. This directory will be created in the container to be used if it does not exist. In this example, enter photos.

Verifying the file transfer

Configuring tAzureStorageList

Procedure

1. Double-click tAzureStorageList to open its Component view.

2. Select the component whose connection details will be used to set up the Azure storage connection. In this example, it is tAzureStorageConnection_1.

3. In the Container name field, enter the name of the container in which you need to check whether the given files exist. In this scenario, it is talendcontainer.

4. Under the Blob filter table, click the [+ ] button to add one row in the table.

5. In the Prefix column, enter the common prefix of the names of the files (blobs) to be checked. This prefix represents a virtual directory level you designate as the starting point down from which files (blobs) are checked. In this example, it is photos/.


6. In the Include sub-directories column, select the check box in the newly added row. This allows tAzureStorageList to check all the files at any hierarchical level beneath the designated starting point.

Configuring tJava

Procedure

1. Double-click tJava to open its Component view.
2. In the Code field, enter `System.out.println();`

3. In the Outline panel, which, by default, is found to the left side of the Component view, expand the `tAzureStorageList` node.

4. From the Outline panel, drop the `CONTAINER_BLOB` global variable into the parentheses in the code in the Component view so as to make the code read: `System.out.println(((Boolean)globalMap.get("tAzureStorageList_1_CURRENT_BLOB")))`;

**Retrieving selected files**

**Procedure**

1. Double-click `tAzureStorageGet` to open its Component view.

2. Select the component whose connection details will be used to set up the Azure storage connection. In this example, it is `tAzureStorageConnection_1`. 
3. In the **Container name** field, enter the name of the container from which you need to retrieve files. In this scenario, it is `talendcontainer`.

4. In the **Local folder** field, enter the path, or browse, to the directory where you want to put the retrieved files. In this example, it is `E:/screenshots`.

5. Under the **Blob** table, click the [+ ] button to add one row in the table.

6. In the **Prefix** column, enter the common name prefix of the files (blobs) to be retrieved. In this example, it is `photos/mongodb/`.

7. In the **Include sub-directories** column, select the check box in the newly added row. This allows `tAzureStorageGet` to retrieve all the files (blobs) beneath the `photos/mongodb/` level.

8. In the **Create parent directories** column, select the check box in the newly added row to create the same directory in the specified local folder as the retrieved blobs have in the container.

   Note that having this same directory is necessary for successfully retrieving blobs. If you leave this check box clear, then you need to create the same directory yourself in the target local folder.

### Executing the Job

#### Procedure

1. Press **F6** to run this Job.
2. Check the execution result on the **Run** console.

   ![Execution console](image)

   You can read that the Job returns the list of the blobs with the `photos` prefix in the container.

3. Double-check the result in the web console of the Azure storage account.
4. Check the retrieved files in the specified local folder.

You can see the blobs with the `photos/mongodb/` prefix have been retrieved and their prefix transformed to directories.
tAzureStorageInputTable

Retrieves a set of entities that satisfy the specified filter criteria from an Azure storage table.

**tAzureStorageInputTable Standard properties**

These properties are used to configure tAzureStorageInputTable running in the Standard Job framework.

The Standard tAzureStorageInputTable component belongs to the Cloud family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
</tbody>
</table>

| Connection Component           | Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list. |

| Account Name                   | Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account. |

| Account Key                    | Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access. |

| Protocol                       | Select the protocol for this connection to be created. |

| Use Azure Shared Access Signature | Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS). |
In the Azure Shared Access Signature field displayed, enter your account SAS URL between double quotation marks. You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is https://<$storagename>.<$service>.core.windows.net/<$sastoken>, where <$storagename> is the storage account name, <$service> is the allowed service name (blob, file, queue or table), and <$sastoken> is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

### Table name

Specify the name of the table from which the entities will be retrieved.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

The schema of this component is predefined with the following columns that describe the three system properties of each entity:

- **PartitionKey**: the partition key for the partition that the entity belongs to.
- **RowKey**: the row key for the entity within the partition.

PartitionKey and RowKey are string type values that uniquely identify every entity in a table, and the user must include them in every insert, update, and delete operation.

- **Timestamp**: the time that the entity was last modified. This DateTime value is maintained by the Azure server and it can not be modified by the user.

For more information about these system properties, see Understanding the Table Service Data Model.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
• Change to built-in property: choose this option to change the schema to Built-in for local changes.

• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

Use filter expression

Select this check box and complete the Filter expressions table displayed to specify the conditions used to filter the entities to be retrieved by clicking the [+] button to add as many rows as needed, each row for a condition, and setting the value for the following parameters for each condition.

• Column: specify the name of the property on which you want to apply for the condition.

• Function: click the cell and select the comparison operator you want to use from the drop-down list.

• Value: specify the value used to compare the property to.

• Predicate: select the predicate used to combine the conditions.

• Field type: click the cell and select the type of the column from the drop-down list.

The generated filter expression will be displayed in the read-only Effective filter field.

For more information about the filter expressions, see Querying Tables and Entities.

Die on error

Select the check box to stop the execution of the Job when an error occurs.

Advanced settings

Name mappings

Complete this table to map the column name of the component schema with the property name of the Azure table entity if they are different.

• Schema column name: enter the column name of the component schema between double quotation marks.

• Entity property name: enter the property name of the Azure table entity between double quotation marks.

For example, if there are three schema columns CompanyID, EmployeeID, and EmployeeName that are used to feed the values for the PartitionKey, RowKey, and Name entity properties respectively, since the PartitionKey and RowKey columns have already been added to the schema automatically and you do not need to specify the mapping relationship for them, you only need to add one row and set the value of the
Schema column name cell with "EmployeeName" and the value of the Entity property name cell with "Name" to specify the mapping relationship for the EmployeeName column when retrieving data from the Azure table.

Global variables

<table>
<thead>
<tr>
<th>Global variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component is usually used as a start component of a Job or Subjob and it always needs an output link.</td>
<td></td>
</tr>
</tbody>
</table>

Handling data with Microsoft Azure Table storage

Here is an example of using Talend components to connect to a Microsoft Azure storage account that gives you access to Azure storage table service, write some employee data into an Azure storage table, and then retrieve that information from the table and display it on the console.

The employee data used in this example is as follows:

```
#Id;Name;Site;Job;Date;Salary
12000;Gerald Roosevelt;Beijing;Software Developer;2008-01-01;15000.01
12001;Benjamin Harrison;Paris;Software Developer;2008-11-22;13000.11
12002;Bob Clinton;Beijing;Software Tester;2008-05-12;12000.22
12003;James Quincy;Paris;Technical Writer;2009-03-10;12000.33
12004;Gerald Harrison;Beijing;Software Tester;2009-06-20;12500.44
12005;Harry Madison;Paris;Software Developer;2009-10-15;14000.55
12006;Helen Roosevelt;Beijing;Software Tester;2009-03-25;13500.66
12007;Mary Clinton;Beijing;Software Developer;2010-02-20;16000.77
12008;Cathey Quincy;Paris;Software Developer;2010-07-15;14000.88
12009;John Smith;Beijing;Technical Writer;2011-02-10;12500.99
```

Creating a Job for handling data with Azure Table storage

Create a Job to connect to an Azure storage account, write some employee data into an Azure storage table, and then retrieve that information from the table and display it on the console.
Procedure

1. Create a new Job and add a `tAzureStorageConnection` component, a `tFixedFlowInput` component, a `tAzureStorageOutputTable` component, a `tAzureStorageInputTable` component, and a `tLogRow` component by typing their names in the design workspace or dropping them from the Palette.
2. Link the `tFixedFlowInput` component to the `tAzureStorageOutputTable` component using a Row > Main connection.
3. Do the same to link the `tAzureStorageInputTable` component to the `tLogRow` component.
4. Link the `tAzureStorageConnection` component to the `tFixedFlowInput` component using a Trigger > OnSubjobOk connection.
5. Do the same to link the `tFixedFlowInput` component to the `tAzureStorageInputTable` component.

Connecting to an Azure Storage account

Configure the `tAzureStorageConnection` component to open the connection to an Azure Storage account.

Before you begin

The Azure Storage account, which allows you to access the Azure Table storage service and store the provided employee data, has already been created. For more information about how to create an Azure Storage account, see About Azure storage accounts.

Procedure

1. Double-click the `tAzureStorageConnection` component to open its Basic settings view on the Component tab.
2. In the **Account Name** field, specify the name of the storage account you need to access.

3. In the **Account Key** field, specify the key associated with the storage account you need to access.

**Writing data into an Azure Storage table**

Configure the **tFixedFlowInput** component and the **tAzureStorageOutputTable** component to write the employee data into an Azure Storage table.

**Procedure**

1. Double-click the **tFixedFlowInput** component to open its **Basic settings** view on the **Component** tab.

2. Click **next to Edit schema** to open the schema dialog box and define the schema by adding six columns: **Id**, **Name**, **Site**, and **Job** of String type, **Date** of Date type, and **Salary** of Double type. Then click **OK** to save the changes and accept the propagation prompted by the pop-up dialog box.
Note that in this example, the **Site** and **Id** columns are used to feed the values of the **PartitionKey** and **RowKey** system properties of each entity and they should be of String type, and the **Name** column is used to feed the value of the **EmployeeName** property of each entity.

3. In the **Mode** area, select **Use Inline Content(delimited file)** and in the **Content** field displayed, enter the employee data that will be written into the Azure Storage table.

4. Double-click the **tAzureStorageOutputTable** component to open its **Basic settings** view on the **Component** tab.

5. From the connection component drop-down list, select the component whose connection details will be used to set up the connection to the Azure Storage service, **tAzureStorageConnection_1** in this example.

6. In the **Table name** field, enter the name of the table into which the employee data will be written, **employee** in this example.

7. From the **Action on table** drop-down list, select the operation to be performed on the specified table, **Drop table if exist and create** in this example.

8. Click **Advanced settings** to open its view.
Click under the **Name mappings** table to add three rows and map the schema column name with the property name of each entity in the Azure table. In this example,

- the **Site** column is used to feed the value of the **PartitionKey** system property, in the first row you need to set the **Schema column name** cell with the value "Site" and the **Entity property name** cell with the value "PartitionKey".

- the **Id** column is used to feed the value of the **RowKey** system property, in the second row you need to set the **Schema column name** cell with the value "Id" and the **Entity property name** cell with the value "RowKey".

- the **Name** column is used to feed the value of the **EmployeeName** property, in the third row you need to set the **Schema column name** cell with the value "Name" and the **Entity property name** cell with the value "EmployeeName".

### Retrieving data from the Azure Storage table

Configure the **tAzureStorageInputTable** component and the **tLogRow** component to retrieve the employee data from the Azure Storage table.

**Procedure**

1. Double-click the **tAzureStorageInputTable** component to open its **Basic settings** view.

2. From the connection component drop-down list, select the component whose connection details will be used to set up the connection to the Azure Storage service, **tAzureStorageConnection_1** in this example.

3. In the **Table name** field, enter the name of the table from which the employee data will be retrieved, **employee** in this example.

4. Click next to **Edit schema** to open the schema dialog box.
Note that the schema has already been predefined with two read-only columns `RowKey` and `PartitionKey` of String type, and another column `Timestamp` of Date type. The `RowKey` and `PartitionKey` columns correspond to the `Id` and `Site` columns of the `tAzureStorageOutputTable` schema.

5. Define the schema by adding another four columns that hold other employee data, `Name` and `Job` of String type, `Date` of Date type, and `Salary` of Double type. Then click OK to save the changes and accept the propagation prompted by the pop-up dialog box.

6. Click **Advanced settings** to open its view.

7. Click **Name mappings** table to add one row and set the **Schema column name** cell with the value "Name" and the **Entity property name** cell with the value "EmployeeName" to map the schema column name with the property name of each entity in the Azure table.

Note that for the `tAzureStorageInputTable` component, the `PartitionKey` and `RowKey` columns have already been added automatically to the schema and you do not need to specify the mapping relationship for them.

8. Double-click the `tLogRow` component to open its **Basic settings** view and in the **Mode** area, select **Table (print values in cells of a table)** for a better display of the result.
Executing the Job to handle data with Azure Table storage

After setting up the Job and configuring the components used in the Job for handling data with Azure Table storage, you can then execute the Job and verify the Job execution result.

Procedure

1. Press \textbf{Ctrl + S} to save the Job.
2. Press \textbf{F6} to execute the Job.

\begin{verbatim}
[statistics] connecting to socket on port 3587
[statistics] connected

<table>
<thead>
<tr>
<th>RowKey</th>
<th>PartitionKey</th>
<th>Timestamp</th>
<th>Name</th>
<th>Job</th>
<th>Data</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>12001</td>
<td>Beijing</td>
<td>2017-03-29</td>
<td>Gerald Roosevelt</td>
<td>Software Developer</td>
<td>2000-01-01</td>
<td>15000.01</td>
</tr>
<tr>
<td>12002</td>
<td>Beijing</td>
<td>2017-03-29</td>
<td>Bob Clinton</td>
<td>Software Tester</td>
<td>2000-05-12</td>
<td>12000.22</td>
</tr>
<tr>
<td>12003</td>
<td>Beijing</td>
<td>2017-03-29</td>
<td>Gerald Harrison</td>
<td>Software Tester</td>
<td>2000-06-21</td>
<td>12500.44</td>
</tr>
<tr>
<td>12004</td>
<td>Beijing</td>
<td>2017-03-29</td>
<td>Helen Roosevelt</td>
<td>Software Tester</td>
<td>2000-03-25</td>
<td>13500.66</td>
</tr>
<tr>
<td>12005</td>
<td>Beijing</td>
<td>2017-03-29</td>
<td>Mary Clinton</td>
<td>Software Developer</td>
<td>2010-02-20</td>
<td>16000.77</td>
</tr>
<tr>
<td>12006</td>
<td>Beijing</td>
<td>2017-03-29</td>
<td>John Smith</td>
<td>Technical Writer</td>
<td>2011-02-10</td>
<td>12500.99</td>
</tr>
<tr>
<td>12007</td>
<td>Paris</td>
<td>2017-03-29</td>
<td>Benjamin Harrison</td>
<td>Software Developer</td>
<td>2000-11-22</td>
<td>13000.11</td>
</tr>
<tr>
<td>12008</td>
<td>Paris</td>
<td>2017-03-29</td>
<td>James Quincy</td>
<td>Technical Writer</td>
<td>2009-03-10</td>
<td>12000.33</td>
</tr>
<tr>
<td>12009</td>
<td>Paris</td>
<td>2017-03-29</td>
<td>Mary Madison</td>
<td>Software Developer</td>
<td>2009-10-15</td>
<td>14000.55</td>
</tr>
<tr>
<td>12010</td>
<td>Paris</td>
<td>2017-03-29</td>
<td>Cathy Quincy</td>
<td>Software Developer</td>
<td>2010-07-15</td>
<td>14000.88</td>
</tr>
</tbody>
</table>

[statistics] disconnected
\end{verbatim}

As shown above, the Job is executed successfully and the employee data is displayed on the console, with the timestamp value that indicates when each entity was inserted.

3. Double-check the employee data that has been written into the Azure Storage table \texttt{employee} using Microsoft Azure Storage Explorer if you want.
tAzureStorageList

Lists blobs in a given container according to the specified blob filters.

**tAzureStorageList Standard properties**

These properties are used to configure tAzureStorageList running in the Standard Job framework.

The Standard tAzureStorageList component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account Name</td>
<td>Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account.</td>
</tr>
<tr>
<td>Account Key</td>
<td>Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Select the protocol for this connection to be created.</td>
</tr>
<tr>
<td>Use Azure Shared Access Signature</td>
<td>Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see <a href="link">Using Shared Access Signatures (SAS)</a>.</td>
</tr>
</tbody>
</table>

In the Azure Shared Access Signature field displayed, enter your account SAS URL between double quotation marks. You can get the SAS URL for each allowed service.
on Microsoft Azure portal after generating SAS. The SAS URL format is `https://<$storagename>.<service>.core.windows.net/<$sastoken>`, where `<$storagename>` is the storage account name, `<$service>` is the allowed service name (blob, file, queue or table), and `<$sastoken>` is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

<table>
<thead>
<tr>
<th>Container name</th>
<th>Enter the name of the container from which you need to select blobs to be listed.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Blob filter</strong></td>
<td>Complete this table to select the blobs to be listed. The parameters to be provided are:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Prefix</strong>: enter the common prefix of the names of the blobs you need to list. This prefix allows you to filter the blobs which have the specified prefix in their names in the given container.</td>
</tr>
<tr>
<td></td>
<td>A blob name contains the virtual hierarchy of the blob itself. This hierarchy is a virtual path to that blob and is relative to the container where that blob is stored. For example, in a container named <code>photos</code>, the name of a photo blob might be <code>2014/US/Oakland/Talend.jpg</code>.</td>
</tr>
<tr>
<td></td>
<td>For this reason, when you define a prefix, you are actually designating a directory level as the blob filter, for example, <code>2014/</code> or <code>2014/US/</code>.</td>
</tr>
<tr>
<td></td>
<td>If you want to select the blobs stored directly beneath the container level, that is to say, the blobs without virtual path in their names, remove quotation marks and enter <code>null</code>.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Include sub-directories</strong>: select this check box to select all of the sub-folders and the blobs in those folders beneath the designated directory level. If you leave this check box clear, <code>tAzureStorageList</code> returns only the blobs, if any, directly beneath that directory level.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
The schema of this component is predefined with a single column `BlobName` of String type, which indicates the name of each blob to be listed.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

**Die on error**

Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a **Row > Reject** link.

**Advanced settings**

- **tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

- **CONTAINER**

  The name of the blob container. This is an After variable and it returns a string.

- **CURRENT_BLOB**

  The blob name being processed by this component. This is an After variable and it returns a string.

- **NB_LINE**

  The number of rows processed. This is an After variable and it returns an integer.

- **ERROR_MESSAGE**

  The error message generated by the component when an error occurs. This is an After variable and it returns a string.

**Usage**

- **Usage rule**

  This component can be used as a standalone component of a Job or Subjob.

- **Prerequisites**

  Knowledge about Microsoft Azure Storage is required.
Related scenario

For a related scenario, see Scenario: Retrieving files from a Azure Storage container on page 285
tAzureStorageOutputTable

Performs the defined action on a given Azure storage table and inserts, replaces, merges or deletes entities in the table based on the incoming data from the preceding component.

tAzureStorageOutputTable Standard properties

These properties are used to configure tAzureStorageOutputTable running in the Standard Job framework.

The Standard tAzureStorageOutputTable component belongs to the Cloud family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td>• Built-In</td>
<td>The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td>• Repository</td>
<td>The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td>Connection Component</td>
<td>Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list.</td>
</tr>
<tr>
<td>Account Name</td>
<td>Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account.</td>
</tr>
<tr>
<td>Account Key</td>
<td>Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Select the protocol for this connection to be created.</td>
</tr>
<tr>
<td>Use Azure Shared Access Signature</td>
<td>Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS).</td>
</tr>
</tbody>
</table>
In the **Azure Shared Access Signature** field displayed, enter your account SAS URL between double quotation marks. You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is https://<$storagename>.<$service>.core.windows.net/<$sastoken>, where <$storagename> is the storage account name, <$service> is the allowed service name (blob, file, queue or table), and <$sastoken> is the SAS token value. For more information, see [Constructing the Account SAS URI](#).

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

<table>
<thead>
<tr>
<th>Table name</th>
<th>Specify the name of the table into which the entities will be written.</th>
</tr>
</thead>
</table>

**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](#).
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](#).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Partition Key</th>
<th>Select the schema column that holds the partition key value from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row Key</td>
<td>Select the schema column that holds the row key value from the drop-down list.</td>
</tr>
<tr>
<td>Action on data</td>
<td>Select an action to be performed on data of the table defined.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Insert</strong>: insert a new entity into the table.</td>
</tr>
</tbody>
</table>
• **Insert or replace**: replace an existing entity or insert a new entity if it does not exist. When replace an entity, any properties from the previous entity will be removed if the new entity does not define them.

• **Insert or merge**: merge an existing entity or insert a new entity if it does not exist. When merge an entity, any properties from the previous entity will be retained if the new entity does not define or include them.

• **Merge**: update an existing entity without removing the property value of the previous entity if the new entity does not define its value.

• **Replace**: update an existing entity and remove the property value of the previous entity if the new entity does not define its value.

• **Delete**: delete an existing entity.

For performance reasons, the incoming data is processed in parallel and in random order. Therefore, it is not recommended to perform any order-sensitive data operation (for example, insert or replace) if there are duplicated rows in your data.

#### Action on table

Select an operation to be performed on the table defined.

- **Default**: No operation is carried out.
- **Drop and create table**: The table is removed and created again.
- **Create table**: The table does not exist and gets created.
- **Create table if does not exist**: The table is created if it does not exist.
- **Drop table if exist and create**: The table is removed if it already exists and created again.

#### Process in batch

Select this check box to process the input entities in batch.

Note that the entities to be processed in batch should belong to the same partition group, which means, they should have the same partition key value.

#### Die on error

Select the check box to stop the execution of the Job when an error occurs.

#### Advanced settings

<table>
<thead>
<tr>
<th>Name mappings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete this table to map the column name of the component schema with the property name of the Azure table entity if they are different.</td>
</tr>
<tr>
<td><strong>Schema column name</strong>: enter the column name of the component schema between double quotation marks.</td>
</tr>
</tbody>
</table>
- **Entity property name**: enter the property name of the Azure table entity between double quotation marks.

For example, if there are three schema columns CompanyID, EmployeeID, and EmployeeName that are used to feed the values for the PartitionKey, RowKey, and Name entity properties respectively, then you need to add the following rows for the mapping when writing data into the Azure table.

- the **Schema column name** cell with the value "CompanyID" and the **Entity property name** cell with the value "PartitionKey".

- the **Schema column name** cell with the value "EmployeeID" and the **Entity property name** cell with the value "RowKey".

- the **Schema column name** cell with the value "EmployeeName" and the **Entity property name** cell with the value "Name".

---

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

---

### Global variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong></td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_SUCCESS</strong></td>
<td>The number of rows successfully processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_REJECT</strong></td>
<td>The number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

---

### Usage

**Usage rule**

This component is usually used as an end component of a Job or Subjob and it always needs an input link.

---

### Related scenario

For a related scenario, see [Handling data with Microsoft Azure Table storage](#) on page 295.
tAzureStoragePut

Uploads local files into a given container for an Azure storage account.

tAzureStoragePut Standard properties

These properties are used to configure tAzureStoragePut running in the Standard Job framework.

The Standard tAzureStoragePut component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> The connection details stored centrally in <strong>Repository &gt; Metadata</strong> will be reused by this component. You need to click the [...] button next to it and in the pop-up <strong>Repository Content</strong> dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

This property is not available when other connection component is selected from the **Connection Component** drop-down list.

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list.</th>
</tr>
</thead>
</table>

| Account Name | Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account. |

| Account Key | Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access. |

| Protocol | Select the protocol for this connection to be created. |

| Use Azure Shared Access Signature | Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS). |

In the **Azure Shared Access Signature** field displayed, enter your account SAS URL between double quotation marks. You can get the SAS URL for each allowed service.
on Microsoft Azure portal after generating SAS. The
SAS URL format is `https://<$storagename>.<$service>.core.windows.net/<$sastoken>`,
where `<$storagename>` is the storage account name,
<$service> is the allowed service name (blob, file, queue or table), and <$sastoken> is the SAS
token value. For more information, see Constructing the
Account SAS URI.

Note that the SAS has valid period, you can set the
start time at which the SAS becomes valid and the
expiry time after which the SAS is no longer valid when
generating it, and you need to make sure your SAS is
still valid when running your Job.

**Container name**
Enter the name of the container you need to write files i
n.
This container must exist in the Azure Storage system
you are using.

**Local folder**
Enter the path, or browse to the folder from which you
need to upload files.

**Azure storage folder**
Enter the path to the virtual blob folder in the remote
Azure storage system you want to upload files into.

If you do not put any value in this field but leave this
field as it is with only its default quotation marks,
tAzureStoragePut writes files directly beneath the
container level.

**Use file list**
Select this check box to be able to define file filtering
conditions. Once selecting it, the Files table is displayed.

**Files**
Complete this table to select the files to be uploaded
into Azure. The parameters to be provided are:
- **Filemask**: file names or path to the files to be
  uploaded.
- **New name**: name to give to the files after they are
  uploaded.

**Die on error**
Select the check box to stop the execution of the Job
when an error occurs.

Clear the check box to skip any rows on error and
complete the process for error-free rows. When errors
are skipped, you can collect the rows on error using a
Row > Reject link.

**Advanced settings**

**tStatCatcher Statistics**
Select this check box to gather the Job processing
metadata at the Job level as well as at each component
level.
Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTAINER</td>
<td>The name of the blob container. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>LOCAL_FOLDER</td>
<td>The local directory used in this component. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>REMOTE_FOLDER</td>
<td>The remote directory used in this component. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage rule</td>
<td>This component can be used as a standalone component of a Job or Subjob.</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>Knowledge about Microsoft Azure Storage is required.</td>
</tr>
</tbody>
</table>

Related scenario

For a related scenario, see Scenario: Retrieving files from a Azure Storage container on page 285
tAzureStorageQueueCreate

Creates a new queue under a given Azure storage account.

**tAzureStorageQueueCreate Standard properties**

These properties are used to configure tAzureStorageQueueCreate running in the Standard Job framework.

The Standard tAzureStorageQueueCreate component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Property Type | Select the way the connection details will be set.  
  - **Built-In**: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.  
  - **Repository**: The connection details stored centrally in Repository > Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.  
  This property is not available when other connection component is selected from the Connection Component drop-down list. |
| Connection Component | Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list. |
| Account Name | Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account. |
| Account Key | Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access. |
| Protocol | Select the protocol for this connection to be created. |
| Use Azure Shared Access Signature | Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS).  
  In the Azure Shared Access Signature field displayed, enter your account SAS URL between double quotation |
marks. You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is https://<$storagename>..<$service>.core.windows.net/<$sastoken>, where <$storagename> is the storage account name, <$service> is the allowed service name (blob, file, queue or table), and <$sastoken> is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

**Queue name**
Specify the name of the Azure queue to be created. For more information about the queue naming rules, see Naming Queues and Metadata.

**Die on error**
Select the check box to stop the execution of the Job when an error occurs.

**Advanced settings**
- **tStatCatcher Statistics**
  Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**
- **QUEUE_NAME**
  The name of the Azure queue. This is an After variable and it returns a string.

- **ERROR_MESSAGE**
  The error message generated by the component when an error occurs. This is an After variable and it returns a string.

**Usage**
- **Usage rule**
  This component can be used as a standalone component of a Job or Subjob.

**Related scenario**
No scenario is available for this component yet.
tAzureStorageQueueDelete

Deletes a specified queue permanently under a given Azure storage account.

**tAzureStorageQueueDelete Standard properties**

These properties are used to configure tAzureStorageQueueDelete running in the Standard Job framework.

The Standard tAzureStorageQueueDelete component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

This property is not available when other connection component is selected from the Connection Component drop-down list.

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Account Name</th>
<th>Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Account Key</th>
<th>Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Select the protocol for this connection to be created.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Use Azure Shared Access Signature</th>
<th>Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS).</th>
</tr>
</thead>
</table>

In the **Azure Shared Access Signature** field displayed, enter your account SAS URL between double quotation marks.
marks. You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is https://<$storagename>.$service>.core.windows.net/<$sastoken>, where <$storagename> is the storage account name, <$service> is the allowed service name (blob, file, queue or table), and <$sastoken> is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

**Queue name**
Specify the name of the Azure queue to be deleted.

**Die on error**
Select the check box to stop the execution of the Job when an error occurs.

**Advanced settings**

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUEUE_NAME</td>
<td>The name of the Azure queue. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th>USAGE RULE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component can be used as a standalone component of a Job or Subjob.</td>
<td></td>
</tr>
</tbody>
</table>
tAzureStorageQueueInput

Retrieves one or more messages from the front of an Azure queue.

**tAzureStorageQueueInput Standard properties**

These properties are used to configure tAzureStorageQueueInput running in the Standard Job framework.

The Standard tAzureStorageQueueInput component belongs to the Cloud family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Account Name</th>
<th>Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Account Key</th>
<th>Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Select the protocol for this connection to be created.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Use Azure Shared Access Signature</th>
<th>Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In the <strong>Azure Shared Access Signature</strong> field displayed, enter your account SAS URL between double quotation marks.</td>
</tr>
</tbody>
</table>
marks. You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is `https://$storagename$.<$service>.core.windows.net/$sastoken>`, where `<$storagename>` is the storage account name, `<$service>` is the allowed service name (blob, file, queue or table), and `<$sastoken>` is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

### Queue name

Specify the name of the Azure queue from which the messages will be retrieved.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

The schema of this component is predefined with the following columns:

- **MessageId**: the id of the message.
- **MessageContent**: the body of the message.
- **InsertionTime**: the time when the message was added to the queue.
- **ExpirationTime**: the time when the message will be expired.
- **NextVisibleTime**: the time when the message becomes visible next time.
- **DequeueCount**: the number of times that the message has been dequeued. This value is incremented each time the message is dequeued, but it will not be incremented when the message is peeked.
- **PopReceipt**: the pop receipt value that is required to delete the message.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
• **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th><strong>Number of messages</strong></th>
<th>Enter the number of messages to be retrieved from the specified queue at a time, up to a maximum of 32.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Peek messages</strong></td>
<td>Select this check box to retrieve messages without removing them from the queue and altering the visibility of them. The messages will remain available to other consumers.</td>
</tr>
<tr>
<td><strong>Delete the message while streaming</strong></td>
<td>Select this check box to delete the messages while retrieving them from the queue.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>Select the check box to stop the execution of the Job when an error occurs.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Visibility timeout in seconds</strong></th>
<th>Enter the visibility timeout value (in seconds) relative to the server time. This timeout value is added to the time at which the message is retrieved to determine its <strong>NextVisibleTime</strong> value. The message will not be visible to other consumers for this time interval after it has been retrieved.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th><strong>NB_LINE</strong></th>
<th>The number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>QUEUE_NAME</strong></td>
<td>The name of the Azure queue. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

**Usage**

| **Usage rule** | This component is usually used as a start component of a Job or Subjob and it always needs an output link. |
Related scenario

No scenario is available for this component yet.
tAzureStorageQueueInputLoop

Runs an endless loop to retrieve messages from the front of an Azure queue.

tAzureStorageQueueInputLoop Standard properties

These properties are used to configure tAzureStorageQueueInputLoop running in the Standard Job framework.

The Standard tAzureStorageQueueInputLoop component belongs to the Cloud family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list.</th>
</tr>
</thead>
</table>

| Account Name         | Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account. |

| Account Key          | Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access. |

| Protocol             | Select the protocol for this connection to be created. |

| Use Azure Shared Access Signature | Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS). |

In the **Azure Shared Access Signature** field displayed, enter your account SAS URL between double quotation marks.
marks. You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is https://<$storagename>.<$service>.core.windows.net/<$sastoken>, where <$storagename> is the storage account name, <$service> is the allowed service name (blob, file, queue or table), and <$sastoken> is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

<table>
<thead>
<tr>
<th>Queue name</th>
<th>Specify the name of the Azure queue from which the messages will be retrieved.</th>
</tr>
</thead>
</table>

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

The schema of this component is predefined with the following columns:

- **MessageId**: the id of the message.
- **MessageContent**: the body of the message.
- **InsertionTime**: the time when the message was added to the queue.
- **ExpirationTime**: the time when the message will be expired.
- **NextVisibleTime**: the time when the message becomes visible next time.
- **DequeuexCount**: the number of times that the message has been dequeued. This value is incremented each time the message is dequeued, but it will not be incremented when the message is peeked.
- **PopReceipt**: the pop receipt value that is required to delete the message.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to *Built-in* for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select *No* upon completion and choose this schema metadata again in the [Repository Content] window.

### Number of messages

Enter the number of messages to be retrieved from the specified queue at a time, up to a maximum of 32.

### Loop wait time

Specify the duration (in seconds) for which the loop will wait for the message to arrive in the queue before returning.

### Die on error

Select the check box to stop the execution of the Job when an error occurs.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**NB_LINE**

The number of rows processed. This is an After variable and it returns an integer.

**QUEUE_NAME**

The name of the Azure queue. This is an After variable and it returns a string.

**ERROR_MESSAGE**

The error message generated by the component when an error occurs. This is an After variable and it returns a string.

### Usage

**Usage rule**

This component is usually used as a start component of a Job or Subjob and it always needs an output link.

### Related scenario

No scenario is available for this component yet.
# tAzureStorageQueueList

Returns all queues associated with the given Azure storage account.

## tAzureStorageQueueList Standard properties

These properties are used to configure tAzureStorageQueueList running in the Standard Job framework.

The Standard tAzureStorageQueueList component belongs to the Cloud family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

This property is not available when other connection component is selected from the Connection Component drop-down list.

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account Name</td>
<td>Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account.</td>
</tr>
<tr>
<td>Account Key</td>
<td>Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Select the protocol for this connection to be created.</td>
</tr>
<tr>
<td>Use Azure Shared Access Signature</td>
<td>Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS).</td>
</tr>
</tbody>
</table>

In the **Azure Shared Access Signature** field displayed, enter your account SAS URL between double quotation marks.
You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is `https://<storagename>.<service>.core.windows.net/<sastoken>`, where `<storagename>` is the storage account name, `<service>` is the allowed service name (blob, file, queue or table), and `<sastoken>` is the SAS token value. For more information, see *[Constructing the Account SAS URI]*.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](https://docs.talend.com/)
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](https://docs.talend.com/)

The schema of this component is predefined with one single column `QueueName` that stores the name of each queue to be returned.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Die on error

Select the check box to stop the execution of the Job when an error occurs.

### Advanced settings

- **tStatCatcher Statistics**: Select this check box to gather the Job processing metadata at the Job level as well as at each component level.
Global Variables

<table>
<thead>
<tr>
<th>NUMBER_OF_QUEUES</th>
<th>The number of queues returned. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule | This component is usually used as a start component of a Job or Subjob and it always needs an output link. |

Related scenario

No scenario is available for this component yet.
tAzureStorageQueueOutput

Adds messages to the back of an Azure queue.
Note that this component can only be used with Java 8.

**tAzureStorageQueueOutput Standard properties**

These properties are used to configure tAzureStorageQueueOutput running in the Standard Job framework.
The Standard tAzureStorageQueueOutput component belongs to the Cloud family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td><strong>Connection Component</strong></td>
<td>Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list.</td>
</tr>
<tr>
<td><strong>Account Name</strong></td>
<td>Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account.</td>
</tr>
<tr>
<td><strong>Account Key</strong></td>
<td>Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access.</td>
</tr>
<tr>
<td><strong>Protocol</strong></td>
<td>Select the protocol for this connection to be created.</td>
</tr>
<tr>
<td><strong>Use Azure Shared Access Signature</strong></td>
<td>Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS).</td>
</tr>
</tbody>
</table>
In the **Azure Shared Access Signature** field displayed, enter your account SAS URL between double quotation marks. You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is `https://<$storagename>.<$service>.core.windows.net/<$sastoken>`, where `<$storagename>` is the storage account name, `<$service>` is the allowed service name (blob, file, queue or table), and `<$sastoken>` is the SAS token value. For more information, see [Constructing the Account SAS URI](#).

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

### Queue name
Specify the name of the Azure queue to which the messages will be added.

### Schema and Edit schema
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](#).
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](#).

The schema of this component is predefined with one single column `MessageContent` that stores the body of each message.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

### Die on error
Select the check box to stop the execution of the Job when an error occurs.
Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>The number of messages processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_SUCCESS</td>
<td>The number of messages successfully enqueued. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_REJECT</td>
<td>The number of messages rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule | This component is usually used as an end component of a Job or Subjob and it always needs an input link. |

Related scenario

No scenario is available for this component yet.
tAzureStorageQueuePurge

Purges messages in an Azure queue.

**tAzureStorageQueuePurge Standard properties**

These properties are used to configure tAzureStorageQueuePurge running in the Standard Job framework.

The Standard tAzureStorageQueuePurge component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
</tbody>
</table>

| Connection Component | Select the component whose connection details will be used to set up the connection to Azure storage from the drop-down list. |

| Account Name | Enter the name of the storage account you need to access. A storage account name can be found in the Storage accounts dashboard of the Microsoft Azure Storage system to be used. Ensure that the administrator of the system has granted you the appropriate access permissions to this storage account. |

| Account Key | Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access. |

| Protocol | Select the protocol for this connection to be created. |

| Use Azure Shared Access Signature | Select this check box to use a shared access signature (SAS) to access the storage resources without need for the account key. For more information, see Using Shared Access Signatures (SAS). |
| | In the Azure Shared Access Signature field displayed, enter your account SAS URL between double quotation marks. |
marks. You can get the SAS URL for each allowed service on Microsoft Azure portal after generating SAS. The SAS URL format is https://<$storagename>.<$service>.core.windows.net/<$sastoken>, where <$storagename> is the storage account name, <$service> is the allowed service name (blob, file, queue or table), and <$sastoken> is the SAS token value. For more information, see Constructing the Account SAS URI.

Note that the SAS has valid period, you can set the start time at which the SAS becomes valid and the expiry time after which the SAS is no longer valid when generating it, and you need to make sure your SAS is still valid when running your Job.

### Queue name
Specify the name of the Azure queue in which the messages will be purged.

### Die on error
Select the check box to stop the execution of the Job when an error occurs.

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as a standalone component of a Job or Subjob.</th>
</tr>
</thead>
</table>

### Related scenario

No scenario is available for this component yet.
tBarChart

Generates a bar chart from the input data to ease technical analysis.
tBarChart reads data from an input flow and transforms the data into a bar chart in a PNG image file.

**tBarChart Standard properties**

These properties are used to configure tBarChart running in the Standard Job framework.
The Standard tBarChart component belongs to the Business Intelligence family.
The component in this framework is generally available.

**Basic settings**

| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Note:**
The schema of tBarChart contains three read-only columns named **series** (string), **category** (string), and **value** (integer) respectively, in a fixed order. The data in any extra columns will be only passed to the next component, if any, without being presented in the bar chart.

| **Built-in**: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide. |
| **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide. |

| **Sync columns** | Click to synchronize the output file schema with the input file schema. The Sync function only displays |

---

332
<table>
<thead>
<tr>
<th><strong>Generated image path</strong></th>
<th>Name and path of the output image file.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Chart title</strong></td>
<td>Enter the title of the bar chart to be generated.</td>
</tr>
<tr>
<td><strong>Include legend</strong></td>
<td>Select this check box if you want the bar chart to include a legend, indicating all series in different colors.</td>
</tr>
<tr>
<td><strong>3Dimensions</strong></td>
<td>Select this check box to create an image with 3D effect. By default, this check box is selected and the bars representing the series of each category will be stacked one over another. If this check box is cleared, a 2D image will be created, with the bars displayed one besides another along the category axis.</td>
</tr>
<tr>
<td><strong>Image width</strong> and <strong>Image height</strong></td>
<td>Enter the width and height of the image file, in pixels.</td>
</tr>
<tr>
<td><strong>Category axis name</strong> and <strong>Value axis name</strong></td>
<td>Enter the category axis name and value axis name.</td>
</tr>
<tr>
<td><strong>Foreground alpha</strong></td>
<td>Enter an integer in the range of 0 to 100 to define the transparency of the image. The smaller the number you enter, the more transparent the image will be.</td>
</tr>
<tr>
<td><strong>Plot orientation</strong></td>
<td>Select the plot orientation of the bar chart: <strong>VERTICAL</strong> or <strong>HORIZONTAL</strong>.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **tStat Catcher Statistics** | Select this check box to collect log data at the component level. |

**Global Variables**

| **Global Variables** | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see **Talend Studio User Guide**. |

For further information about variables, see **Talend Studio User Guide**.
Usage

Usage rule

This component is mainly used as Output component. It requires an Input component and Row main link as input.

Scenario: Creating a bar chart from the input data

This scenario describes a Job that reads source data from a CSV file and transforms the data into a bar chart showing a comparison of several large cities. The input file is shown below:

<table>
<thead>
<tr>
<th>City</th>
<th>Population (x1000)</th>
<th>LandArea (km2)</th>
<th>PopulationDensity (people/km2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beijing</td>
<td>10233</td>
<td>1418</td>
<td>7620</td>
</tr>
<tr>
<td>Moscow</td>
<td>10452</td>
<td>1081</td>
<td>9644</td>
</tr>
<tr>
<td>Seoul</td>
<td>10422</td>
<td>605</td>
<td>17215</td>
</tr>
<tr>
<td>Tokyo</td>
<td>8731</td>
<td>617</td>
<td>14151</td>
</tr>
<tr>
<td>Jakarta</td>
<td>8490</td>
<td>664</td>
<td>12738</td>
</tr>
<tr>
<td>New York</td>
<td>8310</td>
<td>789</td>
<td>10452</td>
</tr>
</tbody>
</table>

Because the input file has a different structure than the one required by the tBarChart component, this use case uses the tMap component to adapt the source data to the three-column schema of tBarChart so that a temporary CSV file can be created as the input to the tBarChart component.

Note:

You will usually use the tMap component to adjust the input schema in accordance with the schema structure of the tBarChart component. For more information about how to use the tMap component, see Talend Studio User Guide and tMap on page 1995.

To ensure correct generation of the temporary input file, a pre-treatment subjob is used to delete the temporary file in case it already exists before the main Job is executed; as this temporary file serves this specific Job only, a post-treatment subjob is used to deleted it after the main Job is executed.

Dropping and linking components

Procedure

1. Drop the following components from the Palette to the design workspace: a tPrejob, a tPostjob, two tFileDelete components, two tFileInputDelimited components, a tMap, three tFileOutputDelimited components, and a tBarChart.
2. Connect the tPrejob component to one tFileDelete component using a Trigger > On Component Ok connection, and connect the tPostjob component to the other tFileDelete component using the same type of connection.
3. Connect the first tFileInputDelimited to the tMap component using a Row > Main connection.
4. Connect the tMap component to the first tFileOutputDelimited component using a Row > Main connection, and name the connection Population.
5. Repeat the step above to connect the tMap component to the other two tFileOutputDelimited components using Row > Main connections, and name the connections Area and Density respectively.
6. Connect the section tFileInputDelimited to the tBarChart component using a Row > Main connection.
7. Connect the first **tFileInputDelimited** component to the second **tFileInputDelimited** component using a **Trigger > On Subjob Ok** connection.

8. Relabel the components to best describe their functionality.

**Results**

![Diagram of component connections](image)

**Reading the source data**

**Procedure**

1. Double-click the first **tFileInputDelimited** component, which is labelled *Large_Cities*, to display its **Basic settings** view.

   ![Basic settings view](image)

   - Fill in the **File name** field by browsing to the input file.
   - In the **Header** field, specify the number of header rows. In this use case, you have only one header row.
   - Click **Edit schema** to describe the data structure of the input file. In this use case, the input schema is made of four columns: *City*, *Population*, *Area*, and *Density*. Upon defining the column names and data types, click **OK** to close the schema dialog box.
Adapting the source data to the tBarChart schema

Procedure

1. Double-click the tMap to open the Map Editor.
   You can see an input table on the input panel, row1 in this example, and three empty output tables, named Population, Area, and Density on the output panel.

2. Use the Schema editor to add three columns to each output table: series (string), category (string), and value (integer).

3. In the relevant Expression field of the output tables, enter the text to be presented in the legend area of the bar chart, "Population (x1000 people)", "Land area (km2)", and "Population density (people/km2)" respectively in this example.

4. Drop the City column of the input table onto the category column of each output table.

5. Drop the Population column of the input table onto the value column of the Population table.

6. Drop the Area column of the input table onto the value column of the Area table.

7. Drop the Density column of the input table onto the value column of the Density table.
8. Click **OK** to save the mappings and close the **Map Editor** and propagate the output schemas to the output components.

**Generating the temporary input file**

**Procedure**

1. Double-click the first **tFileOutputDelimited** component to display its **Basic settings** view.

2. In the **File Name** field, define a temporary CSV file to send the mapped data flows to. In this use case, we name this file **Temp.csv**. This file will be used as the input to the **tBarChart** component.

3. Select the **Append** check box.

4. Repeat the steps above to define the properties of the other two **tFileOutputDelimited** components, using exactly the same settings as in the first **tFileOutputDelimited** component.

   **Note:**
   
   Note that the order of output flows from the **tMap** component is not necessarily the actual order of writing data to the target file. To ensure the target file is correctly generated,
delete the file by the same name if it already exists before Job execution and select the Append check box in all the tFileOutputDelimited components in this step.

Configuring bar chart generation

Procedure

1. Double-click the second tFileInputDelimited component, which is labelled Temp_Input, to display its Basic settings view.

2. Fill in the File name field with the path to the temporary input file generated by the tFileOutputDelimited components. In this use case, the temporary input file to the tBarChart is Temp.csv.

3. Double-click the tBarChart component to display its Basic settings view.

4. In the Generated image path field, define the file path of the image file to be generated.
5. In the Chart title field, define a title for the bar chart.
6. Define the category and series axis names.
7. Define the size and transparency degree of the image if needed. In this use case, we simply use the default settings.
8. Click Edit schema to open the schema dialog box.
9. Copy all the columns from the output schema to the input schema by clicking the left-pointing double arrow button. Then, click OK to close the schema dialog box.

Deleting the temporary file

About this task
As the tPrejob and tPostjob components simply trigger the connected subjobs and do not have any settings to define, all you need to do is to define the properties of the two tFileDelete components.

Procedure
1. Double-click the first tFileDelete component to display its Basic settings view.

2. Fill in the File name field with the path to the temporary input file.
   If the Fail on error check box is selected while the pre-treatment subjob fails because of errors such as the file to delete does not exist, this failure will prevent the main subjob from being launched. In this situation, you can clear the Fail on error check box to avoid this interruption.

3. Specify the same file path in the other tFileDelete component.
Executing the Job

Procedure

1. Press Ctrl+S to save your Job.
2. Press F6 to launch it.

A bar chart is generated, showing a graphical comparison of the specified large cities.
tBigQueryBulkExec

Transfers given data to Google BigQuery.

The tBigQueryOutputBulk and tBigQueryBulkExec components are generally used together as parts of a two step process. In the first step, an output file is generated. In the second step, this file is used to feed a dataset. These two steps are fused together in the tBigQueryOutput component, detailed in a separate section. The advantage of using two separate components is that the data can be transformed before it is loaded in the dataset.

This component transfers a given file from Google Cloud Storage to Google BigQuery, or uploads a given file into Google Cloud Storage and then transfers it to Google BigQuery.

**tBigQueryBulkExec Standard properties**

These properties are used to configure tBigQueryBulkExec running in the Standard Job framework. The Standard tBigQueryBulkExec component belongs to the Big Data family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available: • View schema: choose this option to view the schema only. • Change to built-in property: choose this option to change the schema to Built-in for local changes. • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Client ID and Client secret</td>
<td>Paste the client ID and the client secret, both created and viewable on the API Access tab view of the project</td>
</tr>
</tbody>
</table>
hosting the BigQuery service and the Cloud Storage service you need to use.
To enter the client secret, click the [...] button next to the client secret field, and then in the pop-up dialog box enter the client secret between double quotes and click OK to save the settings.

| **Project ID** | Paste the ID of the project hosting the BigQuery service you need to use.
The ID of your project can be found in the URL of the Google API Console, or by hovering your mouse pointer over the name of the project in the BigQuery Browser Tool. |
| **Authorization code** | Paste the authorization code provided by Google for the access you are building.
To obtain the authorization code, you need to execute the Job using this component and when this Job pauses execution to print out an URL address, you navigate to this address to copy the authorization code displayed. |
| **Dataset** | Enter the name of the dataset you need to transfer data to. |
| **Table** | Enter the name of the table you need to transfer data to.
If this table does not exist, select the **Create the table if it doesn’t exist** check box. |
| **Action on data** | Select the action to be performed from the drop-down list when transferring data to the target table. The action may be:
- **Truncate**: it empties the contents of the table and repopulates it with the transferred data.
- **Append**: it adds rows to the existing data in the table.
- **Empty**: it populates the empty table. |
| **Bulk file already exists in Google storage** | Select this check box to reuse the authentication information for Google Cloud Storage connection, then, complete the **File** and the **Header** fields. |
| **Access key and Secret key** | Paste the authentication information obtained from Google for making requests to Google Cloud Storage.
To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.
These keys can be consulted on the Interoperable Access tab view under the Google Cloud Storage tab of the project. |
| **File to upload** | When the data to be transferred to BigQuery is not stored on Google Cloud Storage, browse to, or enter the path to it. |
| **Bucket** | Enter the name of the bucket, the Google Cloud Storage container, which holds the data to be transferred to Google BigQuery. |
| **File** | Enter the directory of the data stored on Google Cloud Storage and to be transferred to BigQuery.  
If the data is not on Google Cloud Storage, this directory is used as the intermediate destination before the data is transferred to BigQuery. |
| **Header** | Set values to ignore the header of the transferred data.  
For example, enter 0 to ignore no rows for the data without header. |
| **Die on error** | This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows. |

**Advanced settings**

| **token properties File Name** | Enter the path to, or browse to the refresh token file you need to use.  
At the first Job execution using the *Authorization code* you have obtained from Google BigQuery, the value in this field is the directory and the name of that refresh token file to be created and used; if that token file has been created and you need to reuse it, you have to specify its directory and file name in this field.  
With only the token file name entered, *Talend Studio* considers the directory of that token file to be the root of the Studio folder.  
For further information about the refresh token, see the manual of Google BigQuery. |
| **Set the field delimiter** | Enter character, string or regular expression to separate fields for the transferred data. |
| **Encoding** | Select the encoding from the list or select *Custom* and define it manually. This field is compulsory for database data handling. |
| **tStatCatcher Statistics** | Select this check box to collect the log data at the component level. |

**Global Variables**

| **Global Variables** | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

| Usage rule | This is a standalone component. |

Related Scenario

For related topic, see Scenario: Writing data in BigQuery on page 354
tBigQueryInput

Performs the queries supported by Google BigQuery.

This component connects to Google BigQuery and performs queries in it.

**tBigQueryInput Standard properties**

These properties are used to configure tBigQueryInput running in the Standard Job framework.

The Standard tBigQueryInput component belongs to the Big Data family.

The component in this framework is generally available.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

| **Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.

| **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.

| **Client ID and Client secret** | Paste the client ID and the client secret, both created and viewable on the API Access tab view of the project hosting the BigQuery service and the Cloud Storage service you need to use.

To enter the client secret, click the [...] button next to the client secret field, and then in the pop-up dialog box enter the client secret between double quotes and click **OK** to save the settings. |
## Project ID
Paste the ID of the project hosting the BigQuery service you need to use.

The ID of your project can be found in the URL of the Google API Console, or by hovering your mouse pointer over the name of the project in the BigQuery Browser Tool.

## Authorization code
Paste the authorization code provided by Google for the access you are building.

To obtain the authorization code, you need to execute the Job using this component and when this Job pauses execution to print out an URL address, you navigate to this address to copy the authorization code displayed.

## Query
Enter the query you need to use.

## Result size
Select the option depending on the volume of the query result.

By default, the Small option is used, but when the query result is larger than the maximum response size, you need to select the Large option.

If the volume of the result is not certain, select Auto.

## Advanced settings

### token properties File Name
Enter the path to, or browse to the refresh token file you need to use.

At the first Job execution using the Authorization code you have obtained from Google BigQuery, the value in this field is the directory and the name of that refresh token file to be created and used; if that token file has been created and you need to reuse it, you have to specify its directory and file name in this field.

With only the token file name entered, Talend Studio considers the directory of that token file to be the root of the Studio folder.

For further information about the refresh token, see the manual of Google BigQuery.

### Advanced Separator (for number)
Select this check box to change the separator used for the numbers.

### Encoding
Select the encoding from the list or select Custom and define it manually. This field is compulsory for database data handling.

### tStatCatcher Statistics
Select this check box to collect the log data at the component level.

## Global Variables

### Global Variables
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After
bigquery input

variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

| Usage rule         | This is an input component. It sends the extracted data to the component that follows it. |

**Scenario: Performing a query in BigQuery**

This scenario uses two components to perform the SELECT query in BigQuery and present the result in the Studio.

The following figure shows the schema of the table, *UScustomer*, we use as example to perform the SELECT query in.
We will select the State records and count the occurrence of each State among those records.

**Linking the components**

**Procedure**

1. In the **Integration** perspective of Studio, create an empty **Job**, named **BigQueryInput** for example, from the **Job Designs** node in the **Repository** tree view.
   For further information about how to create a **Job**, see the **Talend Studio** User Guide.
2. Drop **tBigQueryInput** and **tLogRow** onto the workspace.
3. Connect them using the **Row > Main** link.

**Creating the query**

**Building access to BigQuery**

**Procedure**

1. Double-click **tBigQueryInput** to open its **Component** view.
2. Click **Edit schema** to open the editor.

3. Click the **+** button twice to add two rows and enter the names of your choice for each of them in the **Column** column. In this scenario, they are: **States** and **Count**.

4. Click **OK** to validate these changes and accept the propagation prompted by the pop-up dialog box.

5. Navigate to the Google APIs Console in your web browser to access the Google project hosting the BigQuery and the Cloud Storage services you need to use.

6. Click the **API Access** tab to open its view.

7. In the **Component** view of the Studio, paste Client ID, Client secret and Project ID from the API Access tab view to the corresponding fields, respectively.

**Getting Authorization code**

**Procedure**

1. In the **Run** view of the Studio, click **Run** to execute this Job. The execution will pause at a given moment to print out in the console the URL address used to get the authorization code.

2. Navigate to this address in your web browser and copy the authorization code displayed.

3. In the **Component** view of **tBigQueryInput**, paste the authorization code in the **Authorization Code** field.
Writing the query

Procedure

In the Query field, enter: `select States, count (*) as Count from documentation.USCustomer group by States`.

Executing the Job

About this task

The tLogRow component presents the execution result of the Job. You can configure the presentation mode on its Component view.

To do this, double-click tLogRow to open the Component view and in the Mode area, select the Table (print values in cells of a table) option.

Procedure

To execute this Job, press F6.

Results

Once done, the Run view is opened automatically, where you can check the execution result.
**tBigQueryOutput**

Transfers the data provided by its preceding component to Google BigQuery. This component writes the data it receives in a user-specified directory and transfers the data to Google BigQuery via Google Cloud Storage.

**tBigQueryOutput Standard properties**

These properties are used to configure tBigQueryOutput running in the Standard Job framework. The Standard tBigQueryOutput component belongs to the Big Data family. The component in this framework is generally available.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. |
| **Property type** | **Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide].  
**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide]. |
<p>| <strong>Local filename</strong> | Browse to, or enter the path to the file you want to write the received data in. |
| <strong>Append</strong> | Select this check box to add rows to the existing data in the file specified in <strong>Local filename</strong>. |
| <strong>Client ID and Client secret</strong> | Paste the client ID and the client secret, both created and viewable on the API Access tab view of the project |</p>
<table>
<thead>
<tr>
<th><strong>BigQueryOutput</strong></th>
<th>hosting the BigQuery service and the Cloud Storage service you need to use.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project ID</strong></td>
<td>Paste the ID of the project hosting the BigQuery service you need to use.</td>
</tr>
<tr>
<td></td>
<td>The ID of your project can be found in the URL of the Google API Console, or by hovering your mouse pointer over the name of the project in the BigQuery Browser Tool.</td>
</tr>
<tr>
<td><strong>Authorization code</strong></td>
<td>Paste the authorization code provided by Google for the access you are building.</td>
</tr>
<tr>
<td></td>
<td>To obtain the authorization code, you need to execute the Job using this component and when this Job pauses execution to print out an URL address, you navigate to this address to copy the authorization code displayed.</td>
</tr>
<tr>
<td><strong>Dataset</strong></td>
<td>Enter the name of the dataset you need to transfer data to.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Enter the name of the table you need to transfer data to.</td>
</tr>
<tr>
<td></td>
<td>If this table does not exist, select the Create the table if it doesn't exist check box.</td>
</tr>
<tr>
<td><strong>Action on data</strong></td>
<td>Select the action to be performed from the drop-down list when transferring data to the target table. The action may be:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Truncate</strong>: it empties the contents of the table and repopulates it with the transferred data.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Append</strong>: it adds rows to the existing data in the table.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Empty</strong>: it populates the empty table.</td>
</tr>
<tr>
<td><strong>Access key and Secret key</strong></td>
<td>Paste the authentication information obtained from Google for making requests to Google Cloud Storage.</td>
</tr>
<tr>
<td></td>
<td>To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td></td>
<td>These keys can be consulted on the Interoperable Access tab view under the Google Cloud Storage tab of the project.</td>
</tr>
<tr>
<td><strong>Bucket</strong></td>
<td>Enter the name of the bucket, the Google Cloud Storage container, which holds the data to be transferred to Google BigQuery.</td>
</tr>
<tr>
<td><strong>File</strong></td>
<td>Enter the directory of the data stored on Google Cloud Storage and to be transferred to BigQuery.</td>
</tr>
</tbody>
</table>
If the data is not on Google Cloud Storage, this directory is used as the intermediate destination before the data is transferred to BigQuery.

Note that the name of the directory must be identical with the name of the file specified in the **Local filename** field.

<table>
<thead>
<tr>
<th>Header</th>
<th>Set values to ignore the header of the transferred data. For example, enter 0 to ignore no rows for the data without header and set 1 for the data with header at the first row.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die on error</td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

### Advanced settings

| token properties File Name | Enter the path to, or browse to the refresh token file you need to use.  
At the first Job execution using the **Authorization code** you have obtained from Google BigQuery, the value in this field is the directory and the name of that refresh token file to be created and used; if that token file has been created and you need to reuse it, you have to specify its directory and file name in this field.  
With only the token file name entered, **Talend Studio** considers the directory of that token file to be the root of the Studio folder.  
For further information about the refresh token, see the manual of Google BigQuery. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Separator</td>
<td>Enter character, string or regular expression to separate fields for the transferred data.</td>
</tr>
<tr>
<td>Create directory if not exists</td>
<td>Select this check box to create the directory you defined in the <strong>File</strong> field for Google Cloud Storage, if it does not exist.</td>
</tr>
<tr>
<td>Custom the flush buffer size</td>
<td>Enter the number of rows to be processed before the memory is freed.</td>
</tr>
<tr>
<td>Check disk space</td>
<td>Select this check box to throw an exception during execution if the disk is full.</td>
</tr>
<tr>
<td>Encoding</td>
<td>Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for database data handling.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect the log data at the component level.</td>
</tr>
</tbody>
</table>
Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**

This is an output component used at the end of a Job. It receives data from its preceding component such as tFileInputDelimited, tMap or tMysqlInput.

Scenario: Writing data in BigQuery

This scenario uses two components to write data in Google BigQuery.

![Diagram](image)

Linking the components

**Procedure**

1. In the **Integration** perspective of *Talend Studio*, create an empty Job, named WriteBigQuery for example, from the **Job Designs** node in the **Repository** tree view.
   
   For further information about how to create a Job, see the *Talend Studio User Guide*.

2. Drop **tRowGenerator** and **tBigQueryOutput** onto the workspace.
   
   The **tRowGenerator** component generates the data to be transferred to Google BigQuery in this scenario. In the real-world case, you can use other components such as **tMysqlInput** or **tMap** in the place of **tRowGenerator** to design a sophisticated process to prepare your data to be transferred.

3. Connect them using the **Row > Main** link.
Preparing the data to be transferred

Procedure

1. Double-click tRowGenerator to open its Component view.

2. Click RowGenerator Editor to open the editor.

3. Click three times to add three rows in the Schema table.

4. In the Column column, enter the name of your choice for each of the new rows. For example, fname, lname and States.

5. In the Functions column, select TalendDataGenerator.getFirstName for the fname row, TalendDataGenerator.getLastName for the lname row and TalendDataGenerator.getUsState for the States row.

6. In the Number of Rows for RowGenerator field, enter, for example, 100 to define the number of rows to be generated.

7. Click OK to validate these changes.

Configuring the access to BigQuery and Cloud Storage

Building access to BigQuery

Procedure

1. Double-click tBigQueryOutput to open its Component view.
2. Click **Sync columns** to retrieve the schema from its preceding component.

3. In the **Local filename** field, enter the directory where you need to create the file to be transferred to BigQuery.

4. Navigate to the Google APIs Console in your web browser to access the Google project hosting the BigQuery and the Cloud Storage services you need to use.

5. Click the API Access tab to open its view.

6. In the **Component** view of the Studio, paste Client ID, Client secret and Project ID from the API Access tab view to the corresponding fields, respectively.

7. In the **Dataset** field, enter the dataset you need to transfer data in. In this scenario, it is **documentation**.

   This dataset must exist in BigQuery. The following figure shows the dataset used by this scenario.

   ![Dataset](image)

8. In the **Table** field, enter the name of the table you need to write data in, for example, **UScustomer**. If this table does not exist in BigQuery you are using, select **Create the table if it doesn't exist**.

9. In the **Action on data** field, select the action. In this example, select **Truncate** to empty the contents, if there are any, of target table and to repopulate it with the transferred data.
Building access to Cloud Storage

Procedure

1. Navigate to the Google APIs Console in your web browser to access the Google project hosting the BigQuery and the Cloud Storage services you need to use.
2. Click Google Cloud Storage > Interoperable Access to open its view.
3. In the Component view of the Studio, paste Access key, Access secret from the Interoperable Access tab view to the corresponding fields, respectively.
4. In the Bucket field, enter the path to the bucket you want to store the transferred data in. In this example, it is talend/documentation
   This bucket must exist in the directory in Cloud Storage

5. In the File field, enter the directory where in Google Cloud Storage you receive and create the file to be transferred to BigQuery. In this example, it is gs://talend/documentation/biquery_UScustomer.csv. The file name must be the same as the one you defined in the Local filename field.
   Troubleshooting: if you encounter issues such as Unable to read source URI of the file stored in Google Cloud Storage, check whether you put the same file name in these two fields.
6. Enter 0 in the Header field to ignore no rows in the transferred data.

Getting Authorization code

Procedure

1. In the Run view of Talend Studio, click Run to execute this Job. The execution will pause at a given moment to print out in the console the URL address used to get the authorization code.
2. Navigate to this address in your web browser and copy the authorization code displayed.
3. In the Component view of tBigQueryOutput, paste the authorization code in the Authorization Code field.

Executing the Job

Procedure

Press F6.

Results

Once done, the Run view is opened automatically, where you can check the execution result.
The data is transferred to Google BigQuery.
### Compose Query

```
select name, lname, States from documentation.UScustomer
```

**Run Query**

**Query Results**

<table>
<thead>
<tr>
<th>Row</th>
<th>Name</th>
<th>Lname</th>
<th>States</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Richard</td>
<td>Carter</td>
<td>Delaware</td>
</tr>
<tr>
<td>2</td>
<td>Warren</td>
<td>Truman</td>
<td>New York</td>
</tr>
<tr>
<td>3</td>
<td>Theodore</td>
<td>Wilson</td>
<td>Maryland</td>
</tr>
<tr>
<td>4</td>
<td>Andrew</td>
<td>Coolidge</td>
<td>Alaska</td>
</tr>
<tr>
<td>5</td>
<td>Ronald</td>
<td>Washington</td>
<td>Mississippi</td>
</tr>
</tbody>
</table>
tBigQueryOutputBulk

Creates a .txt or .csv file for the data of large size so that you can process it according to your needs before transferring it to Google BigQuery.

The tBigQueryOutputBulk and tBigQueryBulkExec components are generally used together as parts of a two step process. In the first step, an output file is generated. In the second step, this file is used to feed a dataset. These two steps are fused together in the tBigQueryOutput component, detailed in a separate section. The advantage of using two separate components is that the data can be transformed before it is loaded in the dataset.

This component writes given data into a .txt or .csv file, ready to be transferred to Google BigQuery.

**tBigQueryOutputBulk Standard properties**

These properties are used to configure tBigQueryOutputBulk running in the Standard Job framework.

The Standard tBigQueryOutputBulk component belongs to the Big Data family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. |

| Property type | **Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**. |
| **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**. |
| **File name** | Browse, or enter the path to the .txt or .csv file you need to generate. |
Append | Select the check box to write new data at the end of the existing data. Otherwise, the existing data will be overwritten.

**Advanced settings**

| Field Separator | Enter character, string or regular expression to separate fields for the transferred data. |
| Create directory if not exists | Select this check box to create the directory you defined in the File field for Google Cloud Storage, if it does not exist. |
| Custom the flush buffer size | Enter the number of rows to be processed before the memory is freed. |
| Check disk space | Select this check box to throw an exception during execution if the disk is full. |
| Encoding | Select the encoding from the list or select Custom and define it manually. This field is compulsory for database data handling. |
| tStatCatcher Statistics | Select this check box to collect the log data at the component level. |

**Global Variables**

Global Variables

| Global Variables | NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. |
| | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |
| | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
| | To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. |
| | For further information about variables, see Talend Studio User Guide. |

**Usage**

| Usage rule | This is an output component which needs the data provided by its preceding component. |
Related Scenario

For related topic, see Scenario: Writing data in BigQuery on page 354
tBonitaDeploy

Deploys a specific Bonita process to a Bonita Runtime.

This component configures any Bonita Runtime engine and deploys a specific Bonita process (a .bar file exported from the Bonita solution) to this engine.

**tBonitaDeploy Standard properties**

These properties are used to configure tBonitaDeploy running in the Standard Job framework. The Standard tBonitaDeploy component belongs to the Business family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bonita version</td>
<td>Select a version number for the Bonita Runtime engine.</td>
</tr>
<tr>
<td>Bonita Runtime Environment File</td>
<td>Browse to, or enter the path to the Bonita Runtime environment file.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This field is displayed only when you select Bonita version 5.3.1 from the Bonita version list.</td>
</tr>
<tr>
<td>Bonita Runtime Home</td>
<td>Browse to, or enter the path to the Bonita Runtime environment directory.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This field is displayed only when you select Bonita version 5.6.1 from the Bonita version list.</td>
</tr>
<tr>
<td>Bonita Runtime Jaas File</td>
<td>Browse to, or enter the path to the Bonita Runtime jaas file.</td>
</tr>
<tr>
<td>Bona Runtime logging file</td>
<td>Browse to, or enter the path to the Bonita Runtime logging file.</td>
</tr>
<tr>
<td>Login Module</td>
<td>Type in the name of login module for logging in Bonita Runtime engine which is defined in the Bonita Runtime jaas file.</td>
</tr>
<tr>
<td>Business Archive</td>
<td>Browse to, or enter the path to the Bonita process .bar file you want to use.</td>
</tr>
<tr>
<td>User name</td>
<td>Type in your user name used to log in Bonita studio.</td>
</tr>
<tr>
<td>Password</td>
<td>Type in your password used to log in Bonita studio.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>
### Die on error

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ProcessDefinitionUUID: the identifier number of the process being deployed. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usually used as a stand-alone component.</td>
</tr>
<tr>
<td>To use this component, you have to manually download the Bonita solution you need to use.</td>
</tr>
</tbody>
</table>

### Connections

<table>
<thead>
<tr>
<th>Outgoing links (from this component to another):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger: Run if; On Component Ok; On Component Error, On Subjob Ok, On Subjob Error.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Incoming links (from one component to this one):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger: Run if, On Component Ok, On Component Error, On Subjob Ok, On Subjob Error</td>
</tr>
</tbody>
</table>

For further information regarding connections, see Connection types in Talend Studio User Guide.

### Limitation

| The Bonita Runtime environment file, the Bonita Runtime jaas file and the Bonita Runtime logging file must be all stored on the execution server of the Job using this component. |
Related Scenario

For related topic, see Scenario 1: Executing a Bonita process via a Talend Job on page 369.
**tBonitaInstantiateProcess**

Starts an instance for a specific process deployed in a Bonita Runtime engine.
This component instantiates a process already deployed in a Bonita Runtime engine.

**tBonitaInstantiateProcess Standard properties**

These properties are used to configure tBonitaInstantiateProcess running in the Standard Job framework.

The Standard tBonitaInstantiateProcess component belongs to the Business family.

The component in this framework is generally available.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

The schema of this component is read-only. You can click Edit schema to view the schema.

In this component the schema is related to the Module selected.

**Note:**
The ProcessInstanceUUID column is pre-defined in the schema of this component, reserved for the identifier number of the process instance being created. |
|---|---|

| **Bonita Client Mode** | Select the client mode you want to use to instantiate a Bonita process.

For more information about all the Bonita client modes, see Bonita’s manuals. |
|---|---|

| **URL** | Enter the URL of the Bonita Web application server you need to access for the process instantiation.

This field is available only in the HTTP client mode. |
|---|---|

| **Auth Username and Auth Password** | Enter the authentication details used to connect to the Bonita Web application server as technical user.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

The default authentication information is provided in these fields. For further information about them, see Bonita’s manuals.

These fields are available only in the HTTP client mode. |
|---|---|
| **Bonita version** | Select the version number of the Bonita Runtime engine to be used.  
This field is available only in the Java client mode. |
|-------------------|-------------------------------------------------------------------------------------------------------------|
| **Bonita Runtime Environment File** | Browse to, or enter the path to the Bonita Runtime environment file.  
This field is available only in the Java client mode.  
**Note:**  
This field is displayed only when you select Bonita version 5.3.1 from the Bonita version list. |
| **Bonita Runtime Home** | Browse to, or enter the path to the Bonita Runtime environment directory.  
**Note:**  
This field is displayed only when you select Bonita version 5.6.1 from the Bonita version list. |
| **Bonita Runtime Jaas File** | Browse to, or enter the path to the Bonita Runtime jaas file.  
This field is available only in the Java client mode. |
| **Bonita Runtime logging file** | Browse to, or enter the path to the Bonita Runtime logging file.  
This field is available only in the Java client mode. |
| **Use Process ID** | Select this check box to instantiate an existing process.  
Once checked, the Process definition ID field is activated in which you can enter the Definition ID of this process.  
This field is available only in the Java client mode.  
**Note:**  
The process definition ID is created when the process is deployed into the Bonita Runtime engine. |
| **Process Name and Process Version** | Enter the ID information of a specific process you want to instantiate. This information is used to automatically generate the ID of this process.  
This field is available in both of the Java client mode and the HTTP client mode. |
| **User name** | Type in your user name used to instantiate this process.  
This field is available in both of the Java client mode and the HTTP client mode. |
| **Password** | Type in your password used to instantiate this process.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter... |
the password between double quotes and click **OK** to save the settings.

This field is available only in the Java client mode.

**Die on error**

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

### Global Variables

**Global Variables**

- **ProcessInstanceUUID**: the identifier number of the process instance being created. This is a Flow variable and it returns a string. It can also be retrieved over the **Row** > **Main** output link.

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

### Usage

**Usage rule**

Usually used as a stand-alone component or as an output component.

To use this component, you have to manually download the Bonita solution you need to use.

**Connections**

Outgoing links (from this component to another):

- **Row**: Main (providing the output parameters from this process)

- **Trigger**: Run if; On Component Ok; On Component Error, On Subjob Ok, On Subjob Error.

Incoming links (from one component to this one):

- **Row**: Main (providing the input parameters to this process)
Scenario 1: Executing a Bonita process via a Talend Job

This scenario describes a Job that deploys a Bonita process into the Bonita Runtime and executes this process, in which a personnel request is treated.

The Job in this scenario uses three components.

- **tBonitaDeploy**: this component deploys a Bonita process into the Bonita Runtime.
- **tFixedFlowInput**: this component generates the schema used as execution parameters of this deployed process.
- **tBonitaInstantiateProcess**: this component executes this deployed process.

Before beginning to replicate this schema, prepare your Bonita.bar file. You need to manually export this file from the Bonita system and then deploy it into the Bonita Runtime engine, using, for example, **tBonitaDeploy** as presented later in this scenario. In this scenario, this file is **TEST--4.0.bar**. Once deployed, this process can be checked via the Bonita interface.
Setting up the Job

Procedure
1. Drop tBonitaDeploy, tFixedFlowInput and tBonitaInstantiateProcess onto the design workspace.
2. Right-click tBonitaDeploy and connect tBonitaDeploy to tFixedFlowInput using a Trigger> On Subject Ok connection.
3. Right-click tFixedFlowInput and connect this component to tBonitaInstantiateProcess using a Row > Main connection.

Configuring the deployment of the process

About this task
To replicate this scenario, proceed as follows:

Procedure
1. Double-click tBonitaDeploy to open its Basic settings view.

2. Select Bonita version 5.3.1 from the Bonita version list. The version you select should be in sync with the version number of the Bonita Runtime engine you are using.
3. In the Bonita Runtime Configuration area, browse to the Bonita Runtime variable files. In the Bonita Runtime Environment File field, browse to the bonita-environnement.xml file; in the Bonita Runtime Jaas File field, browse to the jaas-standard.cfg file; in the Bonita Runtime Logging File field, browse to the logging.properties file.
For users based on Bonita version 5.2.3, only the Bonita Runtime Jaas File field and the Bonita Runtime Logging File field need to be filled.

4. In the Business Archive field, browse to the Bonita .bar file that is the process exported from your Bonita system and will be deployed into the Bonita Runtime engine.

5. In the Username and the Password fields, type in your authentication information to connect to your Bonita.

**Configuring the input flow**

**Procedure**

1. Double-click tFixedFlowInput to open its Basic settings view.
2. Click the three-dot button next to **Edit schema** to open the schema editor.

3. Click the plus button to add one row and rename it as **Name**. This name is identical with the parameter set in Bonita to execute the same process. This way, Bonita can recognize this column as valid parameter and read its value to instantiate this process.

4. Click **OK**.

5. In the **Mode** area of the **Basic settings** view, select the **Use inline table** option and click the plus button to add one row in the table.

6. In the inline table, click the added row and type in the person’s name from your personnel between the quotation marks: *ychen*, whose request will be treated by this deployed process.

**Configuring the Basic settings of tBonitalInstantiateProcess**

**Procedure**

1. Double-click **tBonitalInstantiateProcess** to open its **Basic settings** view.
2. Select Bonita version 5.3.1 from the **Bonita version** list. The version you select should be in sync with the version number of the Bonita Runtime engine you are using.

3. In the **Bonita Runtime Configuration** area, browse to the Bonita Runtime variable files. In the **Bonita Runtime Environment file** field, browse to the `bonita-environnement.xml` file; in the **Bonita Runtime Jaas File** field, browse to the `jaas-standard.cfg` file; in the **Bonita Runtime Logging File** field, browse to the `logging.properties` file.

   For users based on Bonita version 5.2.3, only the **Bonita Runtime Jaas File** field and the **Bonita Runtime Logging File** field need to be filled.

   For users based on Bonita version 5.6.1, in the **Bonita Runtime Home** field, browse to the Bonita Runtime environment directory.
4. Select the **Use Process ID** check box to activate the **Process Definition Id** field.

5. In the **Process Definition Id** field, click between the quotation marks and press **Ctrl+space** to open the auto-completion drop-down list containing the available global variables for this Job.

6. Double-click the variable you need to add it between the quotation marks. In this scenario, double-click `tBonitaDeploy_1_ProcessDefinitionUUID`, which retrieves the process definition ID of the process being deployed by `tBonitaDeploy`.

   ![Image](image.png)

   **Note:**
   You can as well clear the **Use Process ID** check box to activate the **Process name** and the **Process version** fields and enter the corresponding information in the two fields. `tBonitaInstantiateProcess` concatenates the process name and the process version you type in to construct the process definition ID.

7. In the **Username** and **Password** fields, enter the username and password to connect to your Bonita.

### Executing the Job

**Procedure**

Press **F6** to run the Job.

**Results**

![Image](image.png)
This process is deployed into the Bonita Runtime and an instance is created for the personnel requests.

**Scenario 2: Outputting the process instance UUID over the Row > Main link**

This scenario deploys a Bonita process into the Bonita Runtime, starts an instance and outputs the process instance UUID via the Row > Main link.

**Linking the components**

**Procedure**

1. Drop `tBonitaDeploy`, `tBonitaInstantiateProcess` and `tLogRow` onto the workspace.
2. Rename `tBonitaDeploy` as `deploy_process`, `tBonitaInstantiateProcess` as `start_instance` and `tLogRow` as `show_instance_uuid`.
3. Link `tBonitaDeploy` to `tBonitaInstantiateProcess` using the OnSubjobOk trigger.
4. Link `tBonitaInstantiateProcess` to `tLogRow` using a Row > Main connection.

**Configuring the components**

**Procedure**

1. Double-click `tBonitaDeploy` to open its Basic settings view.
2. In the Bonita Runtime Jaas File field, specify the path and name of the jaas file.
   In the Bonita Runtime Loggging File field, specify the path and name of the logging file.
   In the Business Archive field, specify the path and name of the Bonita process.
3. In the Username and Password fields, enter the user authentication credentials.
4. Double-click `tBonitalInstantiateProcess` to open its Basic settings view.
5. In the **Bonita Runtime Jaas File** field, specify the path and name of the jaas file. In the **Bonita Runtime Logging File** field, specify the path and name of the logging file.

6. In the **Process Name** and **Process Version** fields, enter the process information.

7. In the **Username** and **Password** fields, enter the user authentication credentials.

8. Double-click **tLogRow** to open its **Basic settings** view.

9. In the **Mode** area, select **Table (print values in cells of a table)** for better display.

### Executing the Job

**Procedure**

1. Press **Ctrl+S** to save the Job.

2. Press **F6** to run the Job.

   ```
   **** Instance Bonita_V523_Process--1.0--6 created ****
<table>
<thead>
<tr>
<th>show_instance_uuid</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProcessInstanceUUID</td>
</tr>
<tr>
<td>Bonita_V523_Process--1.0--6</td>
</tr>
</tbody>
</table>
   
   [statistics] disconnected
   ```

   As shown above, the instance is created and the UUID is output.
tBoxConnection

Creates a Box connection that the other Box components can reuse.
This component creates the connection to a given Box account.

**tBoxConnection Standard properties**

These properties are used to configure tBoxConnection running in the Standard Job framework.
The Standard tBoxConnection component belongs to the Cloud family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Key</td>
<td>Enter the client key required by Box to access the Box API. To obtain the client key and client secret you need to create an account at <a href="https://developers.box.com/">https://developers.box.com/</a> and then create a Box App under the Box account to be used. The client key and client secret can be obtained from the account application settings.</td>
</tr>
<tr>
<td>Client Secret</td>
<td>Enter the client secret required by Box to access the Box API. To obtain the client key and client secret you need to create an account at <a href="https://developers.box.com/">https://developers.box.com/</a> and then create a Box App under the Box account to be used. The client key and client secret can be obtained from the account application settings.</td>
</tr>
<tr>
<td>Access token</td>
<td>Enter the access token required by Box to access a Box account and operate it. For how to get the access token and refresh token, check the Box documentation you can access from <a href="https://developers.box.com/">https://developers.box.com/</a>.</td>
</tr>
<tr>
<td>Refresh Token</td>
<td>Enter the refresh token required by Box to refresh the access token automatically. For how to get the access token and refresh token, check the Box documentation you can access from <a href="https://developers.box.com/">https://developers.box.com/</a>.</td>
</tr>
<tr>
<td>Use HTTP proxy</td>
<td>If you are using a proxy, select this check box and enter the host and port information of that proxy in the corresponding fields that are displayed.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Variables</td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions</td>
</tr>
</tbody>
</table>
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

| Usage rule                  | This component is used standalone as a Subjob to create the Box connection to be used. In a Job design, it is often connected to the other Box components using the **Trigger** links such as **OnSubjobOk** link. |

Related scenario

For a related scenario, see *Scenario: Uploading and downloading files from Box* on page 391.
tBoxCopy

Copies or moves a given folder or file from Box.

**tBoxCopy Standard properties**

These properties are used to configure tBoxCopy running in the Standard Job framework.

The Standard tBoxCopy component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use existing connection</th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connection/Client Key</strong></td>
<td>Enter the client key required by Box to access the Box API. To obtain the client key and client secret you need to create an account at <a href="https://developers.box.com/">https://developers.box.com/</a> and then create a Box App under the Box account to be used. The client key and client secret can be obtained from the account application settings.</td>
</tr>
<tr>
<td><strong>Connection/Client Secret</strong></td>
<td>Enter the client secret required by Box to access the Box API. To obtain the client key and client secret you need to create an account at <a href="https://developers.box.com/">https://developers.box.com/</a> and then create a Box App under the Box account to be used. The client key and client secret can be obtained from the account application settings.</td>
</tr>
<tr>
<td><strong>Connection/Access Token</strong></td>
<td>Enter the access token required by Box to access a Box account and operate it. For how to get the access token and refresh token, check the Box documentation you can access from <a href="https://developers.box.com/">https://developers.box.com/</a>.</td>
</tr>
<tr>
<td><strong>Connection/Refresh Token</strong></td>
<td>Enter the refresh token required by Box to refresh the access token automatically. For how to get the access token and refresh token, check the Box documentation you can access from <a href="https://developers.box.com/">https://developers.box.com/</a>.</td>
</tr>
<tr>
<td><strong>Connection/Use HTTP proxy</strong></td>
<td>If you are using a proxy, select this check box and enter the host and port information of that proxy in the corresponding fields that are displayed.</td>
</tr>
<tr>
<td><strong>Move Directory</strong></td>
<td>Select this check box to move a directory in Box.</td>
</tr>
<tr>
<td><strong>Copy Directory</strong></td>
<td>Select this check box to copy a directory in Box.</td>
</tr>
<tr>
<td><strong>File Name</strong></td>
<td>Enter file name with the path in Box you want to copy.</td>
</tr>
<tr>
<td><strong>Source Directory</strong></td>
<td>This option appears when the <strong>Move Directory</strong> or <strong>Copy Directory</strong> check box is selected. Enter the source directory in Box to be moved or copied.</td>
</tr>
</tbody>
</table>
### Destination Directory
Enter the destination directory in Box where the specified file or directory will be copied or moved.

### Rename
Select this check box to rename the file or directory to be copied. When copying a file, specify the new file name in the **Destination File Name** field. When copying a directory, enter the new directory name in the **New Directory Name** field.

### Remove Source File
Select this check box to remove the source file during the copy action.

### Schema and Edit schema
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word *line* when naming the fields.

Note that the schema of this component is read-only with four columns named *destinationFilePath*, *destinationFileName*, *sourceDirectory*, and *destinationDirectory*.

### Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>DESTINATION_FILENAME</strong>: the destination file name. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>DESTINATION_FILEPATH</strong>: the destination file path. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>SOURCE_DIRECTORY</strong>: the source directory. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>DESTINATION_DIRECTORY</strong>: the destination directory. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press *Ctrl + Space* to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*. 
Usage

| Usage rule | This component is usually used standalone in a Subjob to copy or move data from Box. |

Related scenarios

No scenario is available for the Standard version of this component yet.
tBoxDelete

Removes a given folder or file from Box.
This component connects to a given Box account and removes a specified file or folder.

tBoxDelete Standard properties

These properties are used to configure tBoxDelete running in the Standard Job framework.
The Standard tBoxDelete component belongs to the Cloud family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>Connection/Client Key</td>
<td>Enter the client key required by Box to access the Box API. To obtain the client key and client secret you need to create an account at <a href="https://developers.box.com/">https://developers.box.com/</a> and then create a Box App under the Box account to be used. The client key and client secret can be obtained from the account application settings.</td>
</tr>
<tr>
<td>Connection/Client Secret</td>
<td>Enter the client secret required by Box to access the Box API. To obtain the client key and client secret you need to create an account at <a href="https://developers.box.com/">https://developers.box.com/</a> and then create a Box App under the Box account to be used. The client key and client secret can be obtained from the account application settings.</td>
</tr>
<tr>
<td>Connection/Access Token</td>
<td>Enter the access token required by Box to access a Box account and operate it. For how to get the access token and refresh token, check the Box documentation you can access from <a href="https://developers.box.com/">https://developers.box.com/</a>.</td>
</tr>
<tr>
<td>Connection/Refresh Token</td>
<td>Enter the refresh token required by Box to refresh the access token automatically. For how to get the access token and refresh token, check the Box documentation you can access from <a href="https://developers.box.com/">https://developers.box.com/</a>.</td>
</tr>
<tr>
<td>Connection/Use HTTP proxy</td>
<td>If you are using a proxy, select this check box and enter the host and port information of that proxy in the corresponding fields that are displayed.</td>
</tr>
<tr>
<td>Path</td>
<td>Enter the path on Box pointing to the folder or the file you need to remove.</td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
</tbody>
</table>
Note that the schema of this component is read-only with one column named `filepath`.

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><code>ERROR_MESSAGE</code>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <code>Die on error</code> check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>REMOVED_PATH</code>: the path of the folder or file being deleted on Box. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <code>Ctrl + Space</code> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

**Usage**

| Usage rule | This component is usually used standalone in a Subjob to remove data from Box. |

**Related scenarios**

No scenario is available for the Standard version of this component yet.
**tBoxGet**

Downloads a selected file from a Box account.

This component connects to a given Box account and downloads files to a specified local directory.

**tBoxGet Standard properties**

These properties are used to configure tBoxGet running in the Standard Job framework.

The Standard tBoxGet component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use existing connection</th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connection/Client Key</strong></td>
<td>Enter the client key required by Box to access the Box API. To obtain the client key and client secret you need to create an account at <a href="https://developers.box.com/">https://developers.box.com/</a> and then create a Box App under the Box account to be used. The client key and client secret can be obtained from the account application settings.</td>
</tr>
<tr>
<td><strong>Connection/Client Secret</strong></td>
<td>Enter the client secret required by Box to access the Box API. To obtain the client key and client secret you need to create an account at <a href="https://developers.box.com/">https://developers.box.com/</a> and then create a Box App under the Box account to be used. The client key and client secret can be obtained from the account application settings.</td>
</tr>
<tr>
<td><strong>Connection/Access Token</strong></td>
<td>Enter the access token required by Box to access a Box account and operate it. For how to get the access token and refresh token, check the Box documentation you can access from <a href="https://developers.box.com/">https://developers.box.com/</a>.</td>
</tr>
<tr>
<td><strong>Connection/Refresh Token</strong></td>
<td>Enter the refresh token required by Box to refresh the access token automatically. For how to get the access token and refresh token, check the Box documentation you can access from <a href="https://developers.box.com/">https://developers.box.com/</a>.</td>
</tr>
<tr>
<td><strong>Connection/Use HTTP proxy</strong></td>
<td>If you are using a proxy, select this check box and enter the host and port information of that proxy in the corresponding fields that are displayed.</td>
</tr>
<tr>
<td><strong>Path</strong></td>
<td>Enter the path on Box pointing to the file you need to download.</td>
</tr>
<tr>
<td><strong>Save as file</strong></td>
<td>Select this check box to display the <strong>Save To</strong> field and browse to, or enter the local directory where you want to store the downloaded file. The existing file, if any, is replaced.</td>
</tr>
</tbody>
</table>
### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Note that the schema of this component is read-only with two columns named `fileName` and `content`.

The Schema field is not available when you have selected the **Save as file** check box.

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>FILE_NAME</strong>: the name of the file being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>INPUT_STREAM</strong>: the content of the file being fetched. This is a Flow variable and it returns an InputStream.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used alone or along with other components via the <strong>Iterate</strong> link or a trigger link such as <strong>OnSubjobOK</strong>.</th>
</tr>
</thead>
</table>

### Related scenario

For a related scenario, see *Scenario: Uploading and downloading files from Box* on page 391.
tBoxList

Lists the files stored in a specified directory in Box.

This component reads the file(s) in Box held in the directory you specify and lists the metadata and the contents of that file or those files.

**tBoxList Standard properties**

These properties are used to configure tBoxList running in the Standard Job framework.

The Standard tBoxList component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection/Client Key</td>
<td>Enter the client key required by Box to access the Box API. To obtain the client key and client secret you need to create an account at <a href="https://developers.box.com/">https://developers.box.com/</a> and then create a Box App under the Box account to be used. The client key and client secret can be obtained from the account application settings.</td>
</tr>
<tr>
<td>Connection/Client Secret</td>
<td>Enter the client secret required by Box to access the Box API. To obtain the client key and client secret you need to create an account at <a href="https://developers.box.com/">https://developers.box.com/</a> and then create a Box App under the Box account to be used. The client key and client secret can be obtained from the account application settings.</td>
</tr>
<tr>
<td>Connection/Access Token</td>
<td>Enter the access token required by Box to access a Box account and operate it. For how to get the access token and refresh token, check the Box documentation you can access from <a href="https://developers.box.com/">https://developers.box.com/</a>.</td>
</tr>
<tr>
<td>Connection/Refresh Token</td>
<td>Enter the refresh token required by Box to refresh the access token automatically. For how to get the access token and refresh token, check the Box documentation you can access from <a href="https://developers.box.com/">https://developers.box.com/</a>.</td>
</tr>
<tr>
<td>Connection/Use HTTP proxy</td>
<td>If you are using a proxy, select this check box and enter the host and port information of that proxy in the corresponding fields that are displayed.</td>
</tr>
<tr>
<td>Path</td>
<td>Enter the path pointing to the folder you need to list the files from, or enter the path pointing to the exact file you need to read.</td>
</tr>
<tr>
<td>List type</td>
<td>Select the type of data you need to list from the specified path, <strong>Files</strong>, <strong>Folders</strong>, or <strong>Both</strong>.</td>
</tr>
<tr>
<td>Include subdirectories</td>
<td>Select this check box to list files from any existing subfolders in addition to the files in the directory defined in the Path field.</td>
</tr>
</tbody>
</table>
### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Note that the schema of this component is read-only with six columns named `name`, `path`, `lastModified`, `size`, `id`, and `type`.

### Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

### Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. |
| | NAME: the name of the remote file being processed. This is a Flow variable and it returns a string. |
| | FILE_PATH: the path pointing to the folder or the file being processed on Box. This is a Flow variable and it returns a string. |
| | FILE_DIRECTORY: the directory of the folder or the file being processed on Box. This is a Flow variable and it returns a string. |
| | LAST_MODIFIED: the timestamp of the last modification of the file being processed. This is a Flow variable and it returns a long. |
| | SIZE: the volume of the file being processed. This is a Flow variable and it returns a long. |
| | ID: the ID of the folder or the file being processed on Box. This is a Flow variable and it returns a string. |
| | TYPE: the type of the objects being processed on Box, file or folder. This is a Flow variable and it returns a string. |

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press `Ctrl + Space` to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | This component is typically used standalone. |
Related scenarios

No scenario is available for the Standard version of this component yet.
tBoxPut

Uploads files to a Box account.
This component uploads data to Box from either a local file or a given data flow.

tBoxPut Standard properties

These properties are used to configure tBoxPut running in the Standard Job framework.
The Standard tBoxPut component belongs to the Cloud family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Use existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection/Client Key</td>
<td>Enter the client key required by Box to access the Box API. To obtain the client key and client secret you need to create an account at <a href="https://developers.box.com/">https://developers.box.com/</a> and then create a Box App under the Box account to be used. The client key and client secret can be obtained from the account application settings.</td>
</tr>
<tr>
<td>Connection/Client Secret</td>
<td>Enter the client secret required by Box to access the Box API. To obtain the client key and client secret you need to create an account at <a href="https://developers.box.com/">https://developers.box.com/</a> and then create a Box App under the Box account to be used. The client key and client secret can be obtained from the account application settings.</td>
</tr>
<tr>
<td>Connection/Access Token</td>
<td>Enter the access token required by Box to access a Box account and operate it. For how to get the access token and refresh token, check the Box documentation you can access from <a href="https://developers.box.com/">https://developers.box.com/</a>.</td>
</tr>
<tr>
<td>Connection/Refresh Token</td>
<td>Enter the refresh token required by Box to refresh the access token automatically. For how to get the access token and refresh token, check the Box documentation you can access from <a href="https://developers.box.com/">https://developers.box.com/</a>.</td>
</tr>
<tr>
<td>Connection/Use HTTP proxy</td>
<td>If you are using a proxy, select this check box and enter the host and port information of that proxy in the corresponding fields that are displayed.</td>
</tr>
<tr>
<td>Remote Path</td>
<td>Enter the path pointing to the file you need to write contents in. This file will be created on the fly if it does not exist.</td>
</tr>
<tr>
<td>Replace if Existing</td>
<td>Select this check box to use the uploaded file to replace the existing one.</td>
</tr>
</tbody>
</table>
### Upload mode

Select the upload mode to be used:

- **Upload incoming content as file**: Select this radio button to read data directly from the input flow of the preceding component and write the data into the file specified in the **Remote Path** field.

- **Upload local file**: Select this radio button to upload a locally stored file to Box. In the **File** field that is displayed, you need to enter the path or browse to this file.

- **Expose as OutputStream**: Select this check box to expose the output stream of this component, which can be used by the other components to write the file content. For example, you can use the **Use output stream** feature of the `tFileOutputDelimited` component to feed a given `tBoxPut`'s exposed output stream. For further information, see `tFileOutputDelimited` on page 1110.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Note that the schema of this component is read-only with a single column named `content` and it receives data from the `content` column of its input schema only. This means that you must use a `content` column in the input data flow to carry the data to be uploaded. This type of column is typically provided by the `tFileInputRaw` component. For further information, see `tFileInputRaw` on page 1081.

The **Schema** field is not available when you have selected the **Expose as OutputStream** or the **Upload local file** upload mode.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

### Global Variables

**Global Variables**

`ERROR_MESSAGE`: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press `Ctrl + Space` to access the variable list and choose the variable to use from it.
For further information about variables, see *Talend Studio User Guide*.

**Usage**

| Usage rule | This component is used either standalone in a Subjob to directly upload a local file to Box or as an end component of a Job flow to upload given data being handled in this flow. |

**Scenario: Uploading and downloading files from Box**

In this scenario, a three-component Job consisting of three Subjobs is created to upload a file to Box and then download a file from Box to the local file system.

Before replicating this scenario, you need to create an account at https://developers.box.com/ and then create a Box App under the Box account to be used. For more information about Box App, see https://app.box.com/developers/services/edit/. The client key and client secret can be obtained from the account application settings. For how to get the access token and refresh token, check the Box documentation you can access from https://developers.box.com/.

**Linking the components**

**Procedure**

1. In the **Integration** perspective of the Studio, create an empty Job from the **Job Designs** node in the **Repository** tree view.
   
   For further information about how to create a Job, see *Talend Studio User Guide*.

2. In the workspace, enter the name of the component to be used and select this component from the list that opens. In this scenario, the components are **tBoxConnection**, **tBoxPut** and **tBoxGet**.

3. Connect **tBoxConnection** to **tBoxPut** using the **Trigger > OnSubjobOk** link.

4. Connect **tBoxPut** to **tBoxGet** using the **Trigger > OnSubjobOk** link.

**Configuring the components**

**Procedure**

1. Double-click **tBoxConnection** to open its **Component** view.
2. Enter the client key, client secret, access token and refresh token in double quotation marks in the relevant fields for accessing the Box account.

3. Double-click **tBoxPut** to open its **Component** view.

4. Select the **Use Existing Connection** check box to reuse the connection created by **tBoxConnection**. In the **Remote Path** field, enter the destination path where you want to upload the file. In the **Upload mode** area, select **Upload Local File**. In the **File** field, enter the file path or browse to the file you want to upload.

5. Double-click **tBoxGet** to open its **Component** view.

6. Select the **Use Existing Connection** check box to reuse the connection created by **tBoxConnection**. In the **Path** field, enter the path of the file that you want to download. Select the **Save As File** check box. In the **Save To** field, enter the file path where to save the file on the local file system.

7. Save the Job.

**Executing the Job**

Execute the Job by pressing **F6** or clicking the **Run** button on the **Run** tab.

The local file, **hello.txt** in this example, is uploaded in your Box.
The file *box.txt* from Box is downloaded to the local file system.
tBufferInput

Retrieves data bufferized via a tBufferOutput component, for example, to process it in another subjob. This component retrieves bufferized data in order to process it in a second subjob.

**tBufferInput Standard properties**

These properties are used to configure tBufferInput running in the Standard Job framework.

The Standard tBufferInput component belongs to the Misc family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
• **View schema**: choose this option to view the schema only.  
• **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.  
In the case of tBufferInput, the column position is more important than the column label as this will be taken into account. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: You create the schema and store it locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository, hence can be reused in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>

**Global Variables**

| Global Variables | **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer. |
**Usage**

| Usage rule | This component is the start component of a secondary Job which is triggered automatically at the end of the main Job. |

**Scenario: Retrieving bufferized data**

This scenario describes a Job that retrieves bufferized data from a subjob and displays it on the console.

- Drop the following components from the **Palette** onto the design workspace: **tFileInputDelimited** and **tBufferOutput**.
- Select the **tFileInputDelimited** and on the **Basic Settings** tab of the **Component** view, set the access parameters to the input file.
• In the **File Name** field, browse to the delimited file holding the data to be bufferized.

• Define the **Row** and **Field separators**, as well as the **Header**.

• Click [...] next to the **Edit schema** field to describe the structure of the file.

• Describe the **Schema** of the data to be passed on to the **tBufferOutput** component.

• Select the **tBufferOutput** component and set the parameters on the **Basic Settings** tab of the **Component** view.

• **Note:**
Generally speaking, the schema is propagated from the input component and automatically fed into the **tBufferOutput** schema. But you can also set part of the schema to be bufferized if you want to.

• Drop the **tBufferInput** and **tLogRow** components from the **Palette** onto the design workspace below the subjob you just created.

• Connect **tFileInputDelimited** and **tBufferInput** via a **Trigger > OnSubjobOk** link and connect **tBufferInput** and **tLogRow** via a **Row > Main** link.

• Double-click **tBufferInput** to set its **Basic settings** in the **Component** view.

• In the **Basic settings** view, click [...] next to the **Edit Schema** field to describe the structure of the file.

• Use the schema defined for the **tFileInputDelimited** component and click **OK**.

• The schema of the **tBufferInput** component is automatically propagated to the **tLogRow**. Otherwise, double-click **tLogRow** to display the **Component** view and click **Sync column**.

• Save your Job and press **F6** to execute it.
The standard console returns the data retrieved from the buffer memory.
**tBufferOutput**

Collects data in a buffer in order to access it later via webservice for example.

*tBufferOutput* has been designed to be exported as WebService in order to access data on the web application server directly. For more information, see *Talend Studio User Guide*.

**tBufferOutput Standard properties**

These properties are used to configure *tBufferOutput* running in the Standard Job framework.

The Standard *tBufferOutput* component belongs to the Misc family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository. Click *Edit schema* to make changes to the schema. If the current schema is of the *Repository* type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to *Built-in* for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select *No* upon completion and choose this schema metadata again in the *Repository Content* window.

In the case of the *tBufferOutput*, the column position is more important than the column label as this will be taken into account. |
| Built-in: The schema will be created and stored locally for this component only. Related topic: see *Talend Studio User Guide*. |
| Repository: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job designs. Related topic: see *Talend Studio User Guide*. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an *After* variable and it returns a string. This variable functions |
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | This component is not startable (green background) and it requires an output component. |

### Scenario 1: Buffering data

This scenario describes an intentionally basic Job that bufferizes data in a child job while a parent Job simply displays the bufferized data onto the standard output console. For an example of how to use **tBufferOutput** to access output data directly on the Web application server, see Scenario 2: Buffering output data on the webapp server on page 402.

- Create two Jobs: a first Job (**BufferFatherJob**) runs the second Job and displays its content onto the **Run** console. The second Job (**BufferChildJob**) stores the defined data into a buffer memory.
- On the first Job, drop the following components: **tRunJob** and **tLogRow** from the **Palette** to the **design workspace**.
- On the second Job, drop the following components: **tFileInputDelimited** and **tBufferOutput** the same way.

Let’s set the parameters of the second Job first:
- Select the **tFileInputDelimited** and on the **Basic Settings** tab of the **Component** view, set the access parameters to the input file.
In **File Name**, browse to the delimited file whose data are to be bufferized.

Define the **Row** and **Field separators**, as well as the **Header**.

Describe the **Schema** of the data to be passed on to the **tBufferOutput** component.

Select the **tBufferOutput** component and set the parameters on the **Basic Settings** tab of the **Component** view.

Generally the schema is propagated from the input component and automatically fed into the **tBufferOutput** schema. But you could also set part of the schema to be bufferized if you want to.

Now on the other **Job (BufferFatherJob)** Design, define the parameters of the **tRunJob** component.

Edit the Schema if relevant and select the column to be displayed. The schema can be identical to the bufferized schema or different.

You could also define context parameters to be used for this particular execution. To keep it simple, the default context with no particular setting is used for this use case.
Press **F6** to execute the parent Job. The **tRunJob** looks after executing the child Job and returns the data onto the standard console:

Starting job BufferFatherJob at 10:41 05-02-2008.

This scenario describes a Job that buffers data to be used as a source system by MDM. An MDM process will invoke this Job to retrieve data by looking up the defined elements (agent region values) from the buffered data. The process can then display the retrieved data in the Talend MDM Web User Interface without really saving them in the MDM hub.

### Creating a data buffer Job

**Procedure**

1. Create a Job named **DetermineRegion**.
2. Drop the following components from the **Palette** onto the design workspace: **tJava**, **tFixedFlowInput**, and **tBufferOutput**.
3. Connect **tJava** to **tFixedFlowInput** using a **Trigger > On Component Ok** link.
4. Connect **tFixedFlowInput** to **tBufferOutput** using a **Row > Main** link.

### Configuring the Job to buffer data

**Procedure**

1. In the **Contexts** view, add a new context variable with the **Name** of **xmlInput** and the **Type** of **String**.

In this example, the context variable **xmlInput** of the Job will be specified in the MDM process which wants to invoke this Job.
You can search for further information about MDM processes on Talend Help Center (https://help.talend.com).

If you cannot find the **Contexts** view, go to **Window > Show view > Talend**, and select **Contexts**.

For more information about how to define context variables, see Talend Studio User Guide.

You can search for further information about how to define context variables on Talend Help Center (https://help.talend.com).

2. Double-click the **tJava** component to open its **Component** view, and in the **Code** area, enter the code according to your needs.

   In this example, enter `System.out.println("#############################"+context.xmlInput);`.

3. Double-click the **tFixedFlowInput** component to open its **Component** view.

4. Click the [...] button next to **Edit schema** to open the dialog box and define the schema for the data to be used by the source system.

   In this example, add one new column `col0` of the type **String**.

5. After the schema is defined, click **Yes** in the **Propagate** dialog box to propagate the schema changes to the following component **tBufferOutput**.

6. In the **Number of rows** field, enter **1**.

7. In the **Mode** area, select **Use Single Table** and enter "Paris" in the **Value** column that corresponds to the column `col0` you have defined.

   In this example, the value of the `col0` provides the agent region information to be retrieved by MDM.

8. Double-click the **tBufferOutput** component to open its **Component** view, and then make sure its schema is synchronized with the previous component **tFixedFlowInput**.

9. Run the Job and make sure the execution succeeds.

**Scenario 2: Buffering output data on the webapp server**

This scenario describes a Job that is called as a Webservice and stores the output data in a buffer directly on the server of the Web application. This scenario creates first a Webservice oriented Job with context variables, and next exports the Job as a Webservice.

**Creating a Job**

**Procedure**

1. Drop the following components from the **Palette** onto the design workspace: **tFixedFlowInput** and **tBufferOutput**.

2. Connect **tFixedFlowInput** to **tBufferOutput** using a **Row Main** link.
Creating a context variable

About this task

For this scenario, you will define two context variables: \textit{nb\_lines} and \textit{lastname}. The first variable will set the number of lines the \texttt{tFixedFlowInput} component will generate, and the second one will set the last name to display in the output list. For more information about how to create and use context variables, see \textit{Talend Studio User Guide}.

To define the two context variables:

Procedure

1. Select the \textbf{Contexts} tab view of your Job, and click the [+] button at the bottom of the view to add two variables, respectively \texttt{nb\_lines} of type \texttt{Integer} and \texttt{lastname} of type \texttt{String}.
2. In the \textbf{Value} field for the variables, set the last name to be displayed and the number of lines to be generated, respectively \textit{Ford} and \textit{3} in this example.

![Context variables configuration](image)

Configuring the input data

Procedure

1. In the design workspace, select \texttt{tFixedFlowInput}.
2. Click the \textbf{Component} tab to define the basic settings for \texttt{tFixedFlowInput}.
3. Click the three-dot [\ldots] button next to \textbf{Edit Schema} to describe the data structure you want to create from internal variables. In this scenario, the schema is made of three columns, \textit{now} of type \texttt{Date}, \textit{firstname} of type \texttt{String}, and \textit{lastname} of type \texttt{String}.

![Schema configuration](image)
4. Click **OK** to close the dialog box and accept propagating the changes when prompted by the system. The three defined columns display in the **Values** panel of the **Basic settings** view of **tFixedFlowInput**.

![Image of tFixedFlowInput configuration](image)

5. Click in the **Value** cell of each of the first two defined columns and press **Ctrl+Space** to access the global variable list.

6. From the global variable list, select `TalendDate.getCurrentDate()` and `talendDatagenerator.getFirstName()`, for the **now** and **firstname** columns respectively.

7. Click in the **Value** cell of **lastname** column and press **Ctrl+Space** to access the global variable list.

8. From the global variable list, select `context.lastname`, the context variable you created for the last name column.

![Image of tFixedFlowInput configuration](image)

**Building your Job as a Webservice**

**About this task**

Before building your Job as a Web service, see *Talend Studio User Guide* for more information.

**Procedure**

1. In the **Repository** tree view, right-click on the above created Job and select **Build Job**. The [Build Job] dialog box appears.
2. Click the **Browse...** button to select a directory to archive your Job in.

3. In the **Build type** panel, select the build type you want to use in the Tomcat webapp directory (WAR in this example) and click **Finish**. The **[Build Job]** dialog box disappears.

4. Copy the War folder and paste it in a Tomcat webapp directory.

**Scenario 3: Calling a Job with context variables from a browser**

This scenario describes how to call the Job you created in **Scenario 2: Buffering output data on the webapp server** on page 402 from your browser with/without modifying the values of the context variables.

Type the following URL into your browser: `http://localhost:8080/export_job/services/export_job3?method=runJob` where “export_job” is the name of the webapp directory deployed in Tomcat and “export_job3” is the name of the Job.
Click Enter to execute your Job from your browser.

- <soapenv:Envelope>
  - <soapenv:Body>
    - <runJobReturn xsi:type="ns1:runJobReturn">
      - <ns1:item xsi:type="ns1:ArrayOf_xsd_string">
        <ns1:item xsi:type="xsd:string">31-07-2008</ns1:item>
        <ns1:item xsi:type="xsd:string">William</ns1:item>
        <ns1:item xsi:type="xsd:string">Ford</ns1:item>
      </ns1:item>
      - <ns1:item xsi:type="ns1:ArrayOf_xsd_string">
        <ns1:item xsi:type="xsd:string">31-07-2008</ns1:item>
        <ns1:item xsi:type="xsd:string">Milard</ns1:item>
        <ns1:item xsi:type="xsd:string">Ford</ns1:item>
      </ns1:item>
      - <ns1:item xsi:type="ns1:ArrayOf_xsd_string">
        <ns1:item xsi:type="xsd:string">31-07-2008</ns1:item>
        <ns1:item xsi:type="xsd:string">James</ns1:item>
        <ns1:item xsi:type="xsd:string">Ford</ns1:item>
      </ns1:item>
    </runJobReturn>
  </soapenv:Body>
</soapenv:Envelope>

The Job uses the default values of the context variables: nb_lines and lastname, that is it generates three lines with the current date, first name and Ford as a last name.

You can modify the values of the context variables directly from your browser. To call the Job from your browser and modify the values of the two context variables, type the following URL:


%20 stands for a blank space in the URL language. In the first argument "arg1", you set the value of the context variable to display "MASSY" as last name. In the second argument "arg2", you set the value of the context variable to "2" to generate only two lines.

Click Enter to execute your Job from your browser.
The Job generates two lines with MASSY as last name.

**Scenario 4: Calling a Job exported as Webservice in another Job**

This scenario describes a Job that calls another Job exported as a Webservice using the `tWebServiceInput`. This scenario will call the Job created in **Scenario 2: Buffering output data on the webapp server** on page 402.

- Drop the following components from the **Palette** onto the design workspace: `tWebServiceInput` and `tLogRow`.
- Connect `tWebServiceInput` to `tLogRow` using a **Row Main** link.
- In the design workspace, select `tWebServiceInput`.
- Click the **Component** tab to define the basic settings for `tWebServiceInput`.

```xml
<soapenv:Envelope>
  <soapenv:Body>
    <runJobReturn xsi:type="ns1:runJobReturn">
      <ns1:item xsi:type="ns1:ArrayOf_xsd_string">
        <ns1:item xsi:type="xsd:string">31-07-2008</ns1:item>
        <ns1:item xsi:type="xsd:string">Richard</ns1:item>
        <ns1:item xsi:type="xsd:string">MASSY</ns1:item>
      </ns1:item>
      <ns1:item xsi:type="ns1:ArrayOf_xsd_string">
        <ns1:item xsi:type="xsd:string">31-07-2008</ns1:item>
        <ns1:item xsi:type="xsd:string">Theodore</ns1:item>
        <ns1:item xsi:type="xsd:string">MASSY</ns1:item>
      </ns1:item>
    </runJobReturn>
  </soapenv:Body>
</soapenv:Envelope>
```
• Set the Schema Type to Built-In and click the three-dot […] button next to Edit Schema to describe the data structure you want to call from the exported Job. In this scenario, the schema is made of three columns, now, firstname, and lastname.

![Schema of tWebServiceInput_1](image)

• Click the plus button to add the three parameter lines and define your variables. Click OK to close the dialog box.

• In the WSDL field of the Basic settings view of tWebServiceInput, enter the URL `http://localhost:8080/export_job/services/export_job3?WSDL` where "export_job" is the name of the webapp directory where the Job to call is stored and "export_job3" is the name of the Job itself.

![tWebServiceInput_1](image)

• In the Method name field, enter `runJob`.

• In the Parameters panel, Click the plus button to add two parameter lines to define your context variables.

• Click in the first Value cell to enter the parameter to set the number of generated lines using the following syntax: `--context_param nb_line=3`.

• Click in the second Value cell to enter the parameter to set the last name to display using the following syntax: `--context_param lastname=Ford`.

• Select tLogRow and click the Component tab to display the component view.
• Set the **Basic settings** for the **tLogRow** component to display the output data in a tabular mode. For more information, see **tLogRow** on page 1988.

• Save your Job and press **F6** to execute it.

```
Starting job Call_Webservice_In_Job at 11:12 01/08/2008.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>now</td>
<td>firstname</td>
<td>lastname</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>01-08-2008</td>
<td>Ronald</td>
<td>Ford</td>
</tr>
<tr>
<td>01-08-2008</td>
<td>Woodrow</td>
<td>Ford</td>
</tr>
<tr>
<td>01-08-2008</td>
<td>Franklin</td>
<td>Ford</td>
</tr>
</tbody>
</table>

Job Call_Webservice_In_Job ended at 14:12 01/08/2008. [exit code=0]
```

The system generates three columns with the current date, first name, and last name and displays them onto the log console in a tabular mode.
tCassandraBulkExec

Improves performance during Insert operations to a Cassandra column family.

The tCassandraOutputBulk and tCassandraBulkExec components are generally used together as parts of a two step process. In the first step, an SSTable is generated. In the second step, this SSTable is written into Cassandra. These two steps are fused together in the tCassandraOutputBulkExec component, detailed in a separate section. The advantage of using two separate components is that the data can be transformed before it is loaded into Cassandra.

tCassandraBulkExec writes data from an SSTable into Cassandra.

**tCassandraBulkExec Standard properties**

These properties are used to configure tCassandraBulkExec running in the Standard Job framework. The Standard tCassandraBulkExec component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB Version</strong></td>
<td>Select the Cassandra version you are using.</td>
</tr>
<tr>
<td><strong>Warning:</strong></td>
<td>- For Cassandra 1.1.2, only local visit on Linux is supported.</td>
</tr>
<tr>
<td></td>
<td>- For Cassandra 1.2.2, make sure that the server and the client run on the same operating system.</td>
</tr>
<tr>
<td></td>
<td>- Cassandra 2.0.0 only works with JVM1.7.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Hostname or IP address of the Cassandra server.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of the Cassandra server.</td>
</tr>
<tr>
<td><strong>Required authentication</strong></td>
<td>Select this check box to provide credentials for the Cassandra authentication.</td>
</tr>
<tr>
<td><strong>Warning:</strong></td>
<td>This check box will not appear if you select Cassandra 1.1.2 from the DB Version list.</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>Fill in this field with the username for the Cassandra authentication.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Fill in this field with the password for the Cassandra authentication.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>
**Use configuration file**

Select this check box and in the field that is displayed, enter the path, or browse to `cassandra.yaml`, the main configuration file for Cassandra.

This way, this component can import and directly use the configuration from `cassandra.yaml`, which can contain many advanced Cassandra properties, such as the properties for SSL encryption.

When you need to run your Job in different Cassandra environments, this feature allows your Job to easily switch between the configurations.

For further information about this `cassandra.yaml` file, see [Cassandra configuration](#).

**Keyspace**

Type in the name of the keyspace into which you want to write the SSTable.

**Column family**

Type in the name of the column family into which you want to write the SSTable.

**SSTable directory**

Specify the local directory of the SSTable to be loaded into Cassandra. Note that the complete path to the SSTable will be the local directory appended by the specified keyspace name and column family name.

For example, if you set the local directory to `/home/talend/sstable`, and specify `testk` as the keyspace name and `testc` as the column family name, the complete path to the SSTable will be `/home/talend/sstable/testk/testc/`.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**Global Variables**

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

  A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

  To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

  For further information about variables, see [Talend Studio User Guide](#).
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as a standalone component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Currently, the execution of this component ends the entire Job.</td>
</tr>
</tbody>
</table>

Related scenarios

No scenario is available for the Standard version of this component yet.
tCassandraClose

Disconnects a connection to a Cassandra server so as to release occupied resources.

**tCassandraClose Standard properties**

These properties are used to configure tCassandraClose running in the Standard Job framework.

The Standard tCassandraClose component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

| Component List | Select an active Cassandra connection to be closed. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

**Usage**

| Usage rule | This component is generally used with other Cassandra components, particularly **tCassandraConnection**. |

**Related Scenario**

For a scenario in which **tCassandraClose** is used, see Scenario: Handling data with Cassandra on page 421.
tCassandraConnection

Enables the reuse of the connection it creates to a Cassandra server.
tCassandraConnection opens a connection to a Cassandra server.

**tCassandraConnection Standard properties**

These properties are used to configure tCassandraConnection running in the Standard Job framework.
The Standard tCassandraConnection component belongs to the Big Data and the Databases families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Property type          | Either Built-In or Repository.  
|------------------------|---------------------------------|
|                        | **Built-In**: No property data stored centrally.  
|                        | **Repository**: Select the repository file where the properties are stored.  

| Use existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.  

| DB Version | Select the Cassandra version you are using.  

| API type | This drop-down list is displayed only when you have selected the 2.0 version of Cassandra from the DB version list. From this API type list, you can either select Datastax to use CQL 3 (Cassandra Query Language) with Cassandra, or select Hector to use CQL 2. Note that the Hector API is deprecated for the 2.0 or later version of Cassandra, but it is still available for use in the Studio so that you can be flexible about the version of the query language to be used with Cassandra 2.0.0.  

| Server | Type in the IP address or hostname of the Cassandra server.  

| Port | Type in the listening port number of the Cassandra server.  

| Required authentication | Select this check box to enable the database authentication.  

| Username | Fill in this field with the username for the Cassandra authentication.  

| Password | Fill in this field with the password for the Cassandra authentication.  


To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**Use SSL connection**
Select this check box to enable the SSL or TLS encrypted connection.

Then you need to use the **tSetKeystore** component in the same Job to specify the encryption information.

For further information about **tSetKeystore**, see **tSetKeystore** on page 3532.

Note that the SSL connection is available only for the version 2.4 + of MongoDB.

### Advanced settings

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**ERROR_MESSAGE**:
the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.

**Usage**

**Usage rule**
This component is generally used with other Cassandra components, particularly **tCassandraClose**.

**Related scenario**

For a scenario in which **tCassandraConnection** is used, see **Scenario: Handling data with Cassandra** on page 421.
**tCassandralInput**

Extracts the desired data from a standard or super column family of a Cassandra keyspace so as to apply changes to the data.

tCassandralInput allows you to read data from a Cassandra keyspace and send data in the Talend flow.

**Mapping tables between Cassandra type and Talend data type**

The first of the following two tables presents the mapping relationships between Cassandra type with Cassandra’s new API, Datastax and *Talend* data type.

**Cassandra 2.0 or later versions**

<table>
<thead>
<tr>
<th>Cassandra Type</th>
<th>Talend Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ascii</td>
<td>String; Character</td>
</tr>
<tr>
<td>BigInt</td>
<td>Long</td>
</tr>
<tr>
<td>Blob</td>
<td>Byte[]</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Counter</td>
<td>Long</td>
</tr>
<tr>
<td>Inet</td>
<td>Object</td>
</tr>
<tr>
<td>Int</td>
<td>Integer; Short; Byte</td>
</tr>
<tr>
<td>List</td>
<td>List</td>
</tr>
<tr>
<td>Map</td>
<td>Object</td>
</tr>
<tr>
<td>Set</td>
<td>Object</td>
</tr>
<tr>
<td>Text</td>
<td>String; Character</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Date</td>
</tr>
<tr>
<td>UUID</td>
<td>String</td>
</tr>
<tr>
<td>TimeUUID</td>
<td>String</td>
</tr>
<tr>
<td>VarChar</td>
<td>String; Character</td>
</tr>
<tr>
<td>VarInt</td>
<td>Object</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Float</td>
<td>Float</td>
</tr>
</tbody>
</table>
Cassandra Hector API (for Cassandra versions older than 2.0)

The following table presents the mapping relationships between Cassandra type with the Hector API and Talend data type.

<table>
<thead>
<tr>
<th>Cassandra Type</th>
<th>Talend Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesType</td>
<td>byte[]</td>
</tr>
<tr>
<td>AsciiType</td>
<td>String</td>
</tr>
<tr>
<td>UTF8Type</td>
<td>String</td>
</tr>
<tr>
<td>IntegerType</td>
<td>Object</td>
</tr>
<tr>
<td>Int32Type</td>
<td>Integer</td>
</tr>
<tr>
<td>LongType</td>
<td>Long</td>
</tr>
<tr>
<td>UUIDType</td>
<td>String</td>
</tr>
<tr>
<td>TimeUUIDType</td>
<td>String</td>
</tr>
<tr>
<td>DateType</td>
<td>Date</td>
</tr>
<tr>
<td>BooleanType</td>
<td>Boolean</td>
</tr>
<tr>
<td>FloatType</td>
<td>Float</td>
</tr>
<tr>
<td>DoubleType</td>
<td>Double</td>
</tr>
<tr>
<td>DecimalType</td>
<td>BigDecimal</td>
</tr>
</tbody>
</table>

`tCassandraInput Standard properties`

These properties are used to configure tCassandraInput running in the Standard Job framework.

The Standard tCassandraInput component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Property type | Either Built-In or Repository.  
<pre><code>           | Built-In: No property data stored centrally. |
</code></pre>
<table>
<thead>
<tr>
<th><strong>Use existing connection</strong></th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB Version</strong></td>
<td>Select the Cassandra version you are using.</td>
</tr>
<tr>
<td><strong>API type</strong></td>
<td>This drop-down list is displayed only when you have selected the 2.0 version of Cassandra from the <strong>DB version</strong> list. From this <strong>API type</strong> list, you can either select <strong>Datastax</strong> to use CQL 3 (Cassandra Query Language) with Cassandra, or select <strong>Hector</strong> to use CQL 2. Note that the <strong>Hector</strong> API is deprecated for the 2.0 or later version of Cassandra, but it is still available for use in the Studio so that you can be flexible about the version of the query language to be used with Cassandra 2.0.0. Along with the evolution of the CQL commands, the parameters to be set in the <strong>Basic settings</strong> view varies.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Hostname or IP address of the Cassandra server.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of the Cassandra server.</td>
</tr>
<tr>
<td><strong>Required authentication</strong></td>
<td>Select this check box to provide credentials for the Cassandra authentication. This check box appears only if you do not select the <strong>Use existing connection</strong> check box.</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>Fill in this field with the username for the Cassandra authentication.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Fill in this field with the password for the Cassandra authentication. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Keyspace</strong></td>
<td>Type in the name of the keyspace from which you want to read data.</td>
</tr>
<tr>
<td><strong>Column family</strong></td>
<td>Type in the name of the column family from which you want to read data.</td>
</tr>
</tbody>
</table>
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  - **View schema**: choose this option to view the schema only. |
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Query

Enter the query statements to be used to read data from the Cassandra database.

By default, the query is not case-sensitive. This means that at runtime, the column names you put in the query are always taken in lower case. If you need to make the query case-sensitive, put the column names in double quotation marks.

The [...] button next to this field allows you to generate the sample code that shows what the pre-defined variables are for the data to be read and how these variables can be used.

This feature is available only for the **Datastax** API of Cassandra 2.0 or a later version.

### Column family type

- **Standard**: Column family is of standard type.
- **Super**: Column family is of super type.

### Include key in output columns

Select this check box to include the key of the column family in output columns.

- **Key column**: select the key column from the list.

### Row key type

Select the appropriate **Talend** data type for the row key from the list.

### Row key Cassandra type

Select the corresponding Cassandra type for the row key from the list.

**Warning:**

The value of the **Default** option varies with the selected row key type. For example, if you select **String** from the **Row key type** list, the value of the **Default** option will be **UTF8**.

For more information about the mapping table between Cassandra type and **Talend** data type, see [Mapping tables between Cassandra type and Talend data type](#) on page 416.

### Include super key output columns

Select this check box to include the super key of the column family in output columns.

- **Super key column**: select the desired super key column from the list.

This check box appears only if you select **Super** from the **Column family type** drop-down list.
<table>
<thead>
<tr>
<th><strong>Super column type</strong></th>
<th>Select the type of the super column from the list.</th>
</tr>
</thead>
</table>
| **Super column Cassandra type** | Select the corresponding Cassandra type for the super column from the list.  
For more information about the mapping table between Cassandra type and Talend data type, see Mapping tables between Cassandra type and Talend data type on page 416. |
| **Specify row keys** | Select this check box to specify the row keys of the column family directly. |
| **Row Keys** | Type in the specific row keys of the column family in the correct format depending on the row key type.  
This field appears only if you select the Specify row keys check box. |
| **Key start** | Type in the start row key of the correct data type. |
| **Key end** | Type in the end row key of the correct data type. |
| **Key limit** | Type in the number of rows to be read between the start row key and the end row key. |
| **Specify columns** | Select this check box to specify the column names of the column family directly. |
| **Columns** | Type in the specific column names of the column family in the correct format depending on the column type.  
This field appears only if you select the Specify columns check box. |
| **Columns range start** | Type in the start column name of the correct data type. |
| **Columns range end** | Type in the end column name of the correct data type. |
| **Columns range limit** | Type in the number of columns to be read between the start column and the end column. |

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| **Global Variables** | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions |
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

| Usage rule | This component always needs an output link. |

**Scenario: Handling data with Cassandra**

This scenario applies only to a Talend solution with Big Data.

This scenario describes a simple Job that reads the employee data from a CSV file, writes the data to a Cassandra keyspace, then extracts the personal information of some employees and displays the information on the console.

This scenario requires six components, which are:
• **tCassandraConnection**: opens a connection to the Cassandra server.

• **tFileInputDelimited**: reads the input file, defines the data structure and sends it to the next component.

• **tCassandraOutput**: writes the data it receives from the preceding component into a Cassandra keyspace.

• **tCassandraInput**: reads the data from the Cassandra keyspace.

• **tLogRow**: displays the data it receives from the preceding component on the console.

• **tCassandraClose**: closes the connection to the Cassandra server.

### Dropping and linking the components

**Procedure**

1. Drop the following components from the Palette onto the design workspace: `tCassandraConnection`, `tFileInputDelimited`, `tCassandraOutput`, `tCassandraInput`, `tLogRow` and `tCassandraClose`.

2. Connect `tFileInputDelimited` to `tCassandraOutput` using a Row > Main link.

3. Do the same to connect `tCassandraInput` to `tLogRow`.

4. Connect `tCassandraConnection` to `tFileInputDelimited` using a Trigger > OnSubjobOk link.

5. Do the same to connect `tFileInputDelimited` to `tCassandraInput` and `tCassandraInput` to `tCassandraClose`.

6. Label the components to better identify their functions.

### Configuring the components

#### Opening a Cassandra connection

**Procedure**

1. Double-click the `tCassandraConnection` component to open its Basic settings view in the Component tab.

2. Select the Cassandra version that you are using from the DB Version list. In this example, it is Cassandra 1.1.2.

3. In the Server field, type in the hostname or IP address of the Cassandra server. In this example, it is localhost.

4. In the Port field, type in the listening port number of the Cassandra server.

5. If required, type in the authentication information for the Cassandra connection: Username and Password.
Reading the input data

Procedure

1. Double-click the **tFileInputDelimited** component to open its Component view.

2. Click the [...] button next to the **File Name/Stream** field to browse to the file that you want to read data from. In this scenario, the directory is \D:/Input/Employees.csv. The CSV file contains four columns: id, age, name and ManagerID.

<table>
<thead>
<tr>
<th>id</th>
<th>age</th>
<th>name</th>
<th>ManagerID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>Alex</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>Peter</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>Mark</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>46</td>
<td>Michael</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>Christophe</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>26</td>
<td>Stephane</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>37</td>
<td>Cedric</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>52</td>
<td>Bill</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>43</td>
<td>Jack</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>28</td>
<td>Andrews</td>
<td>4</td>
</tr>
</tbody>
</table>

3. In the **Header** field, enter 1 so that the first row in the CSV file will be skipped.

4. Click **Edit schema** to define the data to pass on to the **tCassandraOutput** component.

Writing data to a Cassandra keyspace

Procedure

1. Double-click the **tCassandraOutput** component to open its Basic settings view in the Component tab.
2. Type in required information for the connection or use the existing connection you have configured before. In this scenario, the Use existing connection check box is selected.

3. In the Keyspace configuration area, type in the name of the keyspace: Employee in this example, and select Drop keyspace if exists and create from the Action on keyspace list.

4. In the Column family configuration area, type in the name of the column family: Employee_Info in this example, and select Drop column family if exists and create from the Action on column family list.

   The Define column family structure check box appears. In this example, clear this check box.

5. In the Action on data list, select the action you want to carry on, Upsert in this example.

6. Click Sync columns to retrieve the schema from the preceding component.

7. Select the key column of the column family from the Key column list. In this example, it is id.

   If needed, select the Include key in columns check box.

**Reading data from the Cassandra keyspace**

**Procedure**

1. Double-click the tCassandraInput component to open its Component view.

2. Type in required information for the connection or use the existing connection you have configured before. In this scenario, the Use existing connection check box is selected.
3. In the **Keyspace configuration** area, type in the name of the keyspace: *Employee* in this example.

4. In the **Column family configuration** area, type in the name of the column family: *Employee_Info* in this example.

5. Select **Edit schema** to define the data structure to be read from the Cassandra keyspace. In this example, three columns *id*, *name* and *age* are defined.

6. If needed, select the **Include key in output columns** check box, and then select the key column of the column family you want to include from the **Key column** list.

7. From the **Row key type** list, select *Integer* because *id* is of integer type in this example.
   
   Keep the **Default** option for the row key Cassandra type because its value will become the corresponding Cassandra type *Int32* automatically.

8. In the **Query configuration** area, select the **Specify row keys** check box and specify the row keys directly. In this example, three rows will be read. Next, select the **Specify columns** check box and specify the column names of the column family directly. This scenario will read three columns from the keyspace: *id*, *name* and *age*.

9. If needed, the **Key start** and the **Key end** fields allow you to define the range of rows, and the **Key limit** field allows you to specify the number of rows within the range of rows to be read.

   Similarly, the **Columns range start** and the **Columns range end** fields allow you to define the range of columns of the column family, and the **Columns range limit** field allows you to specify the number of columns within the range of columns to be read.

### Displaying the information of interest

**Procedure**

1. Double-click the **tLogRow** component to open its **Component** view.

2. In the **Mode** area, select **Table (print values in cells of a table)**.

### Closing the Cassandra connection

**Procedure**

1. Double-click the **tCassandraClose** component to open its **Component** view.
2. Select the connection to be closed from the Component List.

Saving and executing the Job

Procedure

1. Press Ctrl+S to save your Job.
2. Execute the Job by pressing F6 or clicking Run on the Run tab.

The personal information of three employees is displayed on the console.

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Peter</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>Mark</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>Alex</td>
<td>20</td>
</tr>
</tbody>
</table>
tCassandraOutput

Writes data into or deletes data from a column family of a Cassandra keyspace.
tCassandraOutput receives data from the preceding component, and writes data into Cassandra.

tCassandraOutput Standard properties

These properties are used to configure tCassandraOutput running in the Standard Job framework.
The Standard tCassandraOutput component belongs to the Big Data and the Databases families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-In or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored.</td>
</tr>
<tr>
<td>Use existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the Cassandra version you are using.</td>
</tr>
<tr>
<td>API type</td>
<td>This drop-down list is displayed only when you have selected the 2.0 version of Cassandra from the DB version list. From this API type list, you can either select Datastax to use CQL 3 (Cassandra Query Language) with Cassandra, or select Hector to use CQL 2. Note that the Hector API is deprecated for the 2.0 or later version of Cassandra, but it is still available for use in the Studio so that you can be flexible about the version of the query language to be used with Cassandra 2.0.0. Along with the evolution of the CQL commands, the parameters to be set in the Basic settings view varies.</td>
</tr>
<tr>
<td>Host</td>
<td>Hostname or IP address of the Cassandra server.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of the Cassandra server.</td>
</tr>
<tr>
<td>Required authentication</td>
<td>Select this check box to provide credentials for the Cassandra authentication. This check box appears only if you do not select the Use existing connection check box.</td>
</tr>
<tr>
<td>Username</td>
<td>Fill in this field with the username for the Cassandra authentication.</td>
</tr>
</tbody>
</table>
| **Password** | Fill in this field with the password for the Cassandra authentication.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Use SSL** | Select this check box to enable the SSL or TLS encrypted connection.  
Then you need to use the tSetKeystore component in the same Job to specify the encryption information.  
For further information about tSetKeystore, see tSetKeystore on page 3532. |
| **Keyspace** | Type in the name of the keyspace into which you want to write data. |
| **Action on keyspace** | Select the operation you want to perform on the keyspace to be used:  
- **None**: No operation is carried out.  
- **Drop and create keyspace**: The keyspace is removed and created again.  
- **Create keyspace**: The keyspace does not exist and gets created.  
- **Create keyspace if not exists**: A keyspace gets created if it does not exist.  
- **Drop keyspace if exists and create**: The keyspace is removed if it already exists and created again. |
| **Column family** | Type in the name of the keyspace into which you want to write data. |
| **Action on column family** | Select the operation you want to perform on the column family to be used:  
- **None**: no operation is carried out.  
- **Drop and create column family**: the column family is removed and created again.  
- **Create column family**: the column family does not exist and gets created.  
- **Create column family if not exists**: a column family gets created if it does not exist.  
- **Drop column family if exists and create**: the column family is removed if it already exists and created again. |
| **Action on data** | On the data of the table defined, you can perform:  
- **Upsert**: insert the columns if they do not exist or update the existing columns.  
- **Insert**: insert the columns if they do not exist. This action also updates the existing ones. |
**Update**: update the existing columns or add the columns that do not exist. This action does not support the *Counter* Cassandra data type.

**Delete**: remove columns corresponding to the input flow.

Note that the action list varies depending on the *Hector* or *Datastax* API you are using. When the API is *Datastax*, more actions become available.

For more advanced actions, use the *Advanced settings* view.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click *Edit schema* to make changes to the schema. If the current schema is of the *Repository* type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to *Built-in* for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select *No* upon completion and choose this schema metadata again in the *[Repository Content]* window.

### Built-In

You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

### Repository

You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### Sync columns

Click this button to retrieve schema from the previous component connected in the Job.

### Die on error

Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a *Row > Reject* link.
| Column family type | Standard: Column family is of standard type.  
Super: Column family is of super type. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the row key column from the list.</td>
<td></td>
</tr>
<tr>
<td>Select this check box to include row key in columns.</td>
<td></td>
</tr>
</tbody>
</table>
| Select the super column from the list.  
This drop-down list appears only if you select Super from the Column family type drop-down list. |  |
| Select this check box to include the super columns in standard columns. |  |
| Select this check box to delete the row.  
This check box appears only if you select Delete from the Action on data drop-down list. |  |
| Customize the columns you want to delete. |  |
| Select this check box to delete super columns.  
This check box appears only if you select the Delete Row check box. |  |

### Advanced settings

| Batch Size | Number of lines in each processed batch.  
When you are using the Datastax API, this feature is displayed only when you have selected the Use unlogged batch check box. |
| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

### Global Variables

| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. |
|--------------------|-----------------------------------------------------------------------------------|
For further information about variables, see [Talend Studio User Guide](#).

## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage rule</strong></td>
<td>This component is used as an output component and it always needs an incoming link.</td>
</tr>
</tbody>
</table>

**Note that these features are available to the Datastax API only.**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use unlogged batch</strong></td>
<td><strong>Insert if not exists</strong></td>
</tr>
<tr>
<td><strong>Delete if exists</strong></td>
<td>Select this check box to remove from the target table only the rows that have the same records in the incoming flow. This feature is available only to the <strong>Delete</strong> action.</td>
</tr>
</tbody>
</table>

**Use TTL**

Select this check box to write the TTL data in the target table. In the column list that is displayed, you need to select the column to be used as the TTL column. The DB type of this column must be **Int**. This feature is available to the **Insert** action and the **Update** action only.

**Use Timestamp**

Select this check box to write the timestamp data in the target table. In the column list that is displayed, you need to select the column to be used to store the timestamp data. The DB type of this column must be **BigInt**. This feature is available to the following actions: **Insert**, **Update** and **Delete**.

**IF condition**

Add the condition to be met for the **Update** or the **Delete** action to take place. This condition allows you to be more precise about the columns to be updated or deleted.

**Special assignment operation**

Complete this table to construct advanced SET commands of Cassandra to make the **Update** action more specific. For example, add a record to the beginning or a particular position of a given column.

In the **Update column** column of this table, you need to select the column to be updated and then select the operations to be used from the **Operation** column. The following operations are available:

- **Append**: it adds incoming records to the end of the column to be updated. The Cassandra data types it can handle are Counter, List, Set and Map.
• **Prepend**: it adds incoming records to the beginning of the column to be updated. The only Cassandra data type it can handle is List.

• **Remove**: it removes records from the target table when the same records exist in the incoming flow. The Cassandra data types it can handle are Counter, List, Set and Map.

• **Assign based on position/key**: it adds records to a particular position of the column to be updated. The Cassandra data types it can handle are List and Map.

  Once you select this operation, the **Map key/list position** column becomes editable. From this column, you need to select the column to be used as reference to locate the position to be updated.


<table>
<thead>
<tr>
<th>Row key in the List type</th>
<th>Select the column to be used to construct the WHERE clause of Cassandra to perform the <strong>Update</strong> or the <strong>Delete</strong> action on only selected rows. The column(s) to be used in this table should be from the set of the Primary key columns of the Cassandra table.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Delete collection column based on position/key</strong></td>
<td>Select the column to be used as reference to locate the particular row(s) to be removed. This feature is available only to the <strong>Delete</strong> action.</td>
</tr>
</tbody>
</table>

**Related Scenario**

For a scenario in which **tCassandraOutput** is used, see [Scenario: Handling data with Cassandra](#) on page 421.
tCassandraOutputBulk

Prepares an SSTable of large size and processes it according to your needs before loading this SSTable into a column family of a Cassandra keyspace.

The tCassandraOutputBulk and tCassandraBulkExec components are generally used together as parts of a two step process. In the first step, an SSTable is generated. In the second step, this SSTable is written into Cassandra. These two steps are fused together in the tCassandraOutputBulkExec component, detailed in a separate section. The advantage of using two separate components is that the data can be transformed before it is loaded into Cassandra.

tCassandraOutputBulk receives data from the preceding component, and creates an SSTable locally.

tCassandraOutputBulk Standard properties

These properties are used to configure tCassandraOutputBulk running in the Standard Job framework. The Standard tCassandraOutputBulk component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:
• View schema: choose this option to view the schema only.
• Change to built-in property: choose this option to change the schema to Built-in for local changes.
• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |

| Built-In | You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide. |

| Repository | You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. When the schema to be reused has default values that are integers or functions, ensure that these default... |
values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.
You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

| Table type | Select the type of the data model to be used for the table to be created. It can be CQL (actually CQL3) or non-CQL (the legacy thrift-based API of Cassandra before CQL3).

This drop-down list is available only when the DB version you are using is Cassandra 2.0.0. For the Cassandra versions later than 2.0.0, CQL becomes the only model used by this component and so this list is no longer available. |
|---|---|

<table>
<thead>
<tr>
<th>DB Version</th>
<th>Select the Cassandra version you are using.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Host</th>
<th>Hostname or IP address of the Cassandra server.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Port</th>
<th>Listening port number of the Cassandra server.</th>
</tr>
</thead>
</table>

| Required authentication | Select this check box to provide credentials for the Cassandra authentication.

**Warning:**
This check box will not appear if you select Cassandra 1.1.2 from the DB Version list. |
|---|---|

<table>
<thead>
<tr>
<th>Username</th>
<th>Fill in this field with the username for the Cassandra authentication.</th>
</tr>
</thead>
</table>

| Password | Fill in this field with the password for the Cassandra authentication.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
|---|---|

| Use configuration file | Select this check box and in the field that is displayed, enter the path, or browse to cassandra.yaml, the main configuration file for Cassandra.

This way, this component can import and directly use the configuration from cassandra.yaml, which can contain many advanced Cassandra properties, such as the properties for SSL encryption.

When you need to run your Job in different Cassandra environments, this feature allows your Job to easily switch between the configurations.

For further information about this cassandra.yaml file, see Cassandra configuration. |
|---|---|

<p>| Keyspace | Type in the name of the keyspace into which you want to write the SSTable. |</p>
<table>
<thead>
<tr>
<th><strong>Column family</strong></th>
<th>Type in the name of the column family into which you want to write the SSTable.</th>
</tr>
</thead>
</table>
| **Partitioner**   | Select the partitioner which determines how data is distributed across the Cassandra cluster.  
|                   | • **Random**: default partitioner in Cassandra 1.1 and earlier.  
|                   | • **Murmur3**: default partitioner in Cassandra 1.2.  
|                   | • **Order preserving**: not recommended because it assumes keys are UTF8 strings.  
|                   | For more information about the partitioner, see [http://wiki.apache.org/cassandra/Partitioners](http://wiki.apache.org/cassandra/Partitioners). |
| **Schema statement** | Enter the statement to define the schema of the column family to be used or to be created on the fly.  
| | • This statement is a Cassandra prepared statement, which stores query results locally in the **SSTable directory** you define with this component before sending them to the server. For further information about the prepared statements, see [Prepared statements](#).  
| | • A Cassandra column family is a container for a collection of rows of records that have a similar kind. Its schema must contain strictly the same columns as the component schema you have defined, that is to say, the column names and the order of the columns in both the schemas must be identical.  
| | An example of this schema statement is provided in the **Schema statement** field:  
| | `create table ks.tb (id int, name text, birthday timestamp, primary key (id, birthday)) with clustering order by (birthday desc)`  
| | It will create a column family called `tb` containing the `id`, the `name` and the `birthday` columns under the keyspace `ks`.  
| | For further information about a column family, see [Standard column family](#).  
| | This field is available only when the version of your Cassandra database is later than 2.0.0. When it is 2.0.0, it is available only when you have selected **CQL** from the **Table type** drop-down list. |
| **Insert statement** | Enter the statement to instruct how to write the data from the input flow into the columns of the column family to be used.  
| | This statement is a Cassandra prepared statement, which stores query results locally in the **SSTable directory** you define with this component before sending them to the server. For further information about the prepared statements, see [Prepared statements](#). |
An example of this insert statement is provided in the **Insert statement** field:

```
insert into ks.tb (id, name, birthday) values (?, ?, ?)
```

It will write data into the `id`, the `name` and the `birthday` columns, respectively, of a column family called `tb` in the keyspace `ks`. The question marks in the statement are the bind variable markers for the three columns. For further information about bind variables and their usage, see **Bound parameters** and **Parameterized queries**.

This field is available only when the version of your Cassandra database is later than 2.0.0. When it is 2.0.0, it is available only when you have selected **CQL** from the **Table type** drop-down list.

<table>
<thead>
<tr>
<th>Column name comparator</th>
<th>Select the data type for the column names, which is used to sort columns. This list is not available when the data model to be used is CQL3. For more information about the comparators, see <a href="http://www.datastax.com/docs/1.1/ddl/column_family#about-data-types-comparators-and-validators">http://www.datastax.com/docs/1.1/ddl/column_family#about-data-types-comparators-and-validators</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTable directory</td>
<td>Specify the local directory for the SSTable. Note that the complete path to the SSTable will be the local directory appended by the specified keyspace name and column family name. For example, if you set the local directory to <code>/home/talend/sstable</code>, and specify <code>testk</code> as the keyspace name and <code>testc</code> as the column family name, the complete path to the SSTable will be <code>/home/talend/sstable/testk/testc/</code>.</td>
</tr>
<tr>
<td>Buffer size</td>
<td>Specify what size the SSTable must reach before it is written into Cassandra.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>tStatCatcher Statistics</strong></th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</th>
</tr>
</thead>
</table>

For further information about variables, see Talend Studio User Guide.

## Usage

| Usage rule | This component always needs an incoming link. |

## Related scenarios

No scenario is available for the Standard version of this component yet.
tCassandraOutputBulkExec

Improves performance during Insert operations to a column family of a Cassandra keyspace.

The tCassandraOutputBulk and tCassandraBulkExec components are generally used together to output data to an SSTable and then to write the SSTable into Cassandra, in a two step process. These two steps are fused together in the tCassandraOutputBulkExec component.

tCassandraOutputBulkExec receives data from the preceding component, creates an SSTable and then writes the SSTable into Cassandra.

**tCassandraOutputBulkExec Standard properties**

These properties are used to configure tCassandraOutputBulkExec running in the Standard Job framework.

The Standard tCassandraOutputBulkExec component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
• **View schema**: choose this option to view the schema only.  
• **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.  

| Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.  
| Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.  
When the schema to be reused has default values that are integers or functions, ensure that these default values are appropriate. |
values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

**Table type**
Select the type of the data model to be used for the table to be created. It can be **CQL** (actually CQL3) or **non-CQL** (the legacy thrift-based API of Cassandra before CQL3).

This drop-down list is available only when the **DB version** you are using is Cassandra 2.0.0. For the Cassandra versions later than 2.0.0, CQL becomes the only model used by this component and so this list is no longer available.

**DB Version**
Select the Cassandra version you are using.

**Warning:**
- For Cassandra 1.1.2, only local visit on Linux is supported.
- For Cassandra 1.2.2, make sure that the server and the client run on the same operating system.
- Cassandra 2.0.0 only works with JVM1.7.

**Host**
Hostname or IP address of the Cassandra server.

**Port**
Listening port number of the Cassandra server.

**Required authentication**
Select this check box to provide credentials for the Cassandra authentication.

**Warning:**
This check box will not appear if you select Cassandra 1.1.2 from the **DB Version** list.

**Username**
Fill in this field with the username for the Cassandra authentication.

**Password**
Fill in this field with the password for the Cassandra authentication.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

**Keyspace**
Type in the name of the keyspace into which you want to write the SSTable.

**Column family**
Type in the name of the column family into which you want to write the SSTable.

**Partitioner**
Select the partitioner which determines how the data is distributed across the Cassandra cluster.
- **Random**: default partitioner in Cassandra 1.1 and earlier.
- **Murmur3**: default partitioner in Cassandra 1.2.
- **Order preserving**: not recommended because it assumes keys are UTF8 strings.

For more information about the partitioner, see [http://wiki.apache.org/cassandra/Partitioners](http://wiki.apache.org/cassandra/Partitioners).

### Schema statement

Enter the statement to define the schema of the column family to be used or to be created on the fly.

- This statement is a Cassandra prepared statement, which stores query results locally in the **SSTable directory** you define with this component before sending them to the server. For further information about the prepared statements, see **Prepared statements**.

- A Cassandra column family is a container for a collection of rows of records that have a similar kind. Its schema must contain strictly the same columns as the component schema you have defined, that is to say, the column names and the order of the columns in both the schemas must be identical.

An example of this schema statement is provided in the **Schema statement** field:

```sql
create table ks.tb (id int, name text, birthday timestamp, primary key(id, birthday)) with clustering order by (birthday desc)
```

It will create a column family called `tb` containing the `id`, the `name` and the `birthday` columns under the keyspace `ks`.

For further information about a column family, see **Standard column family**.

This field is available only when the version of your Cassandra database is later than 2.0.0. When it is 2.0.0, it is available only when you have selected **CQL** from the **Table type** drop-down list.

### Insert statement

Enter the statement to instruct how to write the data from the input flow into the columns of the column family to be used.

This statement is a Cassandra prepared statement, which stores query results locally in the **SSTable directory** you define with this component before sending them to the server. For further information about the prepared statements, see **Prepared statements**.

An example of this insert statement is provided in the **Insert statement** field:

```sql
insert into ks.tb (id, name, birthday) values (?, ?, ?)
```
It will write data into the id, the name and the birthday columns, respectively, of a column family called tb in the keyspace ks. The question marks in the statement are the bind variable markers for the three columns. For further information about bind variables and their usage, see Bound parameters and Parameterized queries.

This field is available only when the version of your Cassandra database is later than 2.0.0. When it is 2.0.0, it is available only when you have selected CQL from the Table type drop-down list.

<table>
<thead>
<tr>
<th>Column name comparator</th>
<th>Select the data type for the column names, which is used to sort columns. For more information about the comparators, see <a href="http://www.datastax.com/docs/1.1/ddl/column_family#about-data-types-comparators-and-validators">http://www.datastax.com/docs/1.1/ddl/column_family#about-data-types-comparators-and-validators</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTable directory</td>
<td>Specify the local directory for the SSTable. Note that the complete path to the SSTable will be the local directory appended by the specified keyspace name and column family name. For example, if you set the local directory to /home/talend/sstable, and specify testk as the keyspace name and testc as the column family name, the complete path to the SSTable will be /home/talend/sstable/testk/testc/.</td>
</tr>
<tr>
<td>Buffer size</td>
<td>Specify what size the SSTable must reach before it is written into Cassandra.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</th>
</tr>
</thead>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is mainly used when no particular transformation is required on the data to be loaded into the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Currently, the execution of this component ends the entire Job.</td>
</tr>
</tbody>
</table>

Related scenarios

No scenario is available for the Standard version of this component yet.
tCassandraRow

Acts on the actual DB structure or on the data, depending on the nature of the query and the database.

tCassandraRow is the specific component for this database query. It executes the Cassandra Query Language (CQL) query stated in the specified database. The row suffix means the component implements a flow in the Job design although it does not provide output.

**tCassandraRow Standard properties**

These properties are used to configure tCassandraRow running in the Standard Job framework.
The Standard tCassandraRow component belongs to the Big Data and the Databases families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored.</td>
</tr>
<tr>
<td>Use existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the Cassandra version you are using.</td>
</tr>
<tr>
<td>API type</td>
<td>This drop-down list is displayed only when you have selected the 2.0 version of Cassandra from the DB version list. From this API type list, you can either select Datastax to use CQL 3 (Cassandra Query Language) with Cassandra, or select Hector to use CQL 2. Note that the Hector API is deprecated for the 2.0 or later version of Cassandra, but it is still available for use in the Studio so that you can be flexible about the version of the query language to be used with Cassandra 2.0.0.</td>
</tr>
<tr>
<td>Host</td>
<td>Type in the IP address or hostname of the Cassandra server.</td>
</tr>
<tr>
<td>Port</td>
<td>Type in the listening port number of the Cassandra server.</td>
</tr>
<tr>
<td>Required Authentication</td>
<td>Select this check box to provide credentials for the Cassandra authentication. This check box appears only if you do not select the Use existing connection check box.</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>Fill in this field with the username for the Cassandra authentication.</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Fill in this field with the password for the Cassandra authentication.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Keyspace</strong></td>
<td>Type in the name of the keyspace on which you want to execute the CQL commands.</td>
</tr>
<tr>
<td><strong>Column family</strong></td>
<td>Name of the column family.</td>
</tr>
</tbody>
</table>
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
| **Query**         | Type in the CQL command to be executed.  
  By default, the query is not case-sensitive. This means that at runtime, the column names you put in the query are always taken in lower case. If you need to make the query case-sensitive, put the column names in double quotation marks. |
| **Die on error**  | This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows. |

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |
Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide. |

Usage

| Limitation | Currently, the select commands are not supported. |

Related scenario

For related topics, see

- Scenario: Removing and regenerating a MySQL table index on page 2552.
- Scenario: Using PreparedStatement objects to query data on page 2553.
**tCentricCRMInput**

Extracts data from a Centric CRM DB based on a query.

Connects to a module of a Centric CRM database via the relevant Web service.

### tCentricCRMInput Standard properties

These properties are used to configure tCentricCRMInput running in the Standard Job framework.

The Standard tCentricCRMInput component belongs to the Business family.

The component in this framework is generally available.

#### Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CentricCRM URL</strong></td>
<td>Type in the Web service URL to connect to the CentricCRM DB.</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>Select the relevant module in the list</td>
</tr>
<tr>
<td><strong>Server</strong></td>
<td>Type in the IP address of the DB server.</td>
</tr>
</tbody>
</table>
| **UserID** and **Password** | Type in the Web service user authentication data.  
To enter the password, click the [..] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Schema** and **Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.  
In this component the schema is related to the Module selected. |
| **Query condition** | Type in the query to select the data to be extracted.                       |
Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

Global Variables

| Global Variables | NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. |
| | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |
| | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
| | To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. |
| | For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | Usually used as a Start component. An output component is required. |

Related scenarios

No scenario is available for the Standard version of this component yet.
tCentricCRMOutput

Writes data into a CentricCRM DB.
Writes data in a module of a CentricCRM database via the relevant Web service.

**tCentricCRMOutput Standard properties**

These properties are used to configure tCentricCRMOutput running in the Standard Job framework. The Standard tCentricCRMOutput component belongs to the Business family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>CentricCRM URL</th>
<th>Type in the Web service URL to connect to the CentricCRM DB.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>Select the relevant module in the list</td>
</tr>
<tr>
<td>Server</td>
<td>IP address of the DB server</td>
</tr>
<tr>
<td>UserID and Password</td>
<td>Type in the Web service user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Action</td>
<td>Insert, Update or Delete the data in the CentricCRM module.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td></td>
<td>Click Sync columns to retrieve the schema from the previous component connected in the Job.</td>
</tr>
</tbody>
</table>
### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</th>
</tr>
</thead>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Used as an output component. An Input component is required.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>This component requires installation of its related jar files.</td>
</tr>
</tbody>
</table>

### Related scenarios

No scenario is available for the Standard version of this component yet.
tChangeFileEncoding

Transforms the character encoding of a given file and generates a new file with the transformed character encoding.

```
tChangeFileEncoding changes the encoding of a given file.
```

**tChangeFileEncoding Standard properties**

These properties are used to configure `tChangeFileEncoding` running in the Standard Job framework.

The Standard `tChangeFileEncoding` component belongs to the Data Quality and the File families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Custom Input Encoding</td>
<td>Select this check box to customize input encoding type. When it is selected, a list of input encoding types appears, allowing you to select an input encoding type or specify an input encoding type by selecting <code>CUSTOM</code>.</td>
</tr>
<tr>
<td>Encoding</td>
<td>From this list of character encoding types, you can select one of the offered options or customize the character encoding by selecting <code>CUSTOM</code> and specifying a character encoding type.</td>
</tr>
<tr>
<td>Input File Name</td>
<td>Path of the input file.</td>
</tr>
<tr>
<td>Output File Name</td>
<td>Path of the output file.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create directory if does not exist</td>
<td>This check box is selected by default. It creates a directory to hold the output table if required.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

```
| Global Variables | EXIST: the result of whether a specified file exists. This is a Flow variable and it returns a boolean.
|                  | FILENAME: the name of the file processed. This is an After variable and it returns a string.
|                  | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.
|                  | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
```
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

| Usage rule | This component can be used as standalone component. |

### Scenario: Transforming the character encoding of a file

This Java scenario describes a very simple Job that transforms the character encoding of a text file and generates a new file with the new character encoding.

### Procedure

#### Procedure

1. Drop a tChangeFileEncoding component onto the design workspace.

![tChangeFileEncoding component on design workspace](image)

2. Double-click the tChangeFileEncoding component to display its Basic settings view.

![Basic settings view for tChangeFileEncoding](image)

<table>
<thead>
<tr>
<th>Input File Name</th>
<th>Encoding</th>
<th>Output File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;F:\in.txt&quot;</td>
<td>GB2312</td>
<td>&quot;F:\out.txt&quot;</td>
</tr>
</tbody>
</table>

3. Select Use Custom Input Encoding check box. Set the Encoding type to GB2312.
4. In the Input File Name field, enter the file path or browse to the input file.
5. In the Output File Name field, enter the file path or browse to the output file.
6. Select CUSTOM from the second Encoding list and enter UTF-16 in the text field.
7.

Press F6 to execute the Job.

Results
The encoding type of the file *in.txt* is transformed and *out.txt* is generated with the UTF-16 encoding type.
**tChronometerStart**

Operates as a chronometer device that starts calculating the processing time of one or more subjobs in the main Job, or that starts calculating the processing time of part of your subjob.

Starts measuring the time a subjob takes to be executed.

**tChronometerStart Standard properties**

These properties are used to configure tChronometerStart running in the Standard Job framework.

The Standard tChronometerStart component belongs to the Logs & Errors family.

The component in this framework is generally available.

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>STARTTIME: the start time to calculate the processing time of subjob(s). This is a Flow variable and it returns a long.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Variables</td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td>Global Variables</td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td>Global Variables</td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td>Global Variables</td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

**Usage**

| Usage rule | You can use tChronometerStart as a start or middle component. It can precede one or more processing tasks in the subjob. It can precede one or more subjobs in the main Job. |

**Related scenario**

For related scenario, see Scenario: Measuring the processing time of a subjob and part of a subjob on page 455.
tChronometerStop

Operates as a chronometer device that stops calculating the processing time of one or more subjobs in the main Job, or that stops calculating the processing time of part of your subjob. tChronometerStop displays the total execution time.

Measures the time a subjob takes to be executed.

**tChronometerStop Standard properties**

These properties are used to configure tChronometerStop running in the Standard Job framework.

The Standard tChronometerStop component belongs to the Logs & Errors family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Since options</th>
<th>Select either check box to select measurement starting point:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Since the beginning:</strong> stops time measurement launched at the beginning of a subjob.</td>
</tr>
<tr>
<td></td>
<td><strong>Since a tChronometerStart:</strong> stops time measurement launched at one of the tChronometerStart components used on the data flow of the subjob.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Display duration in console</th>
<th>When selected, it displays subjob execution information on the console.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Display component name</th>
<th>When selected, it displays the name of the component on the console.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Caption</th>
<th>Enter desired text, to identify your subjob for example.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Display human readable duration</th>
<th>When selected, it displays subjob execution information in readable time units.</th>
</tr>
</thead>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>STOPTIME:</strong> the stop time to calculate the processing time of subjob(s). This is a Flow variable and it returns a long.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>DURATION:</strong> the processing time of subjob(s). This is a Flow variable and it returns a long.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE:</strong> the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
</tbody>
</table>
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

**Usage**

| Usage rule | Cannot be used as a start component. |

**Scenario: Measuring the processing time of a subjob and part of a subjob**

This scenario is a subjob that does the following in a sequence:

- generates 1000 000 rows of first and last names,
- gathers first names with their corresponding last names,
- stores the output data in a delimited file,
- measures the duration of the subjob as a whole,
- measures the duration of the name replacement operation,
- displays the gathered information about the processing time on the Run log console.

To measure the processing time of the subjob:

- Drop the following components from the Palette onto the design workspace: tRowGenerator, tMap, tFileOutputDelimited, and tChronometerStop.
- Connect the first three components using Main Row links.

**Note:**

When connecting tMap to tFileOutputDelimited, you will be prompted to name the output table. The name used in this example is "new_order".

- Connect tFileOutputDelimited to tChronometerStop using an OnComponentOk link.
- Select tRowGenerator and click the Component tab to display the component view.
• In the component view, click Basic settings. The Component tab opens on the Basic settings view by default.

• Click Edit schema to define the schema of the RowGenerator. For this Job, the schema is composed of two columns: First_Name and Last_Name, so click twice the [+] button to add two columns and rename them.

• Click the RowGenerator Editor three-dot button to open the editor and define the data to be generated.

<table>
<thead>
<tr>
<th>Column</th>
<th>Key</th>
<th>Type</th>
<th>Length</th>
<th>Precision</th>
<th>Decimals</th>
<th>Functions</th>
<th>Preview</th>
</tr>
</thead>
<tbody>
<tr>
<td>First_Name</td>
<td></td>
<td>String</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>getFirstName</td>
<td></td>
</tr>
<tr>
<td>Last_Name</td>
<td></td>
<td>String</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>TalendDataGenerator.getLastName</td>
<td></td>
</tr>
</tbody>
</table>

• In the RowGenerator Editor, specify the number of rows to be generated in the Number of Rows for RowGenerator field and click OK. The RowGenerator Editor closes.

• You will be prompted to propagate changes. Click Yes in the popup message.

• Double-click on the tMap component to open the Map editor. The Map editor opens displaying the input metadata of the RowGenerator component.
In the **Schema editor** panel of the Map editor, click the plus button of the output table to add two rows and define them.

In the Map editor, drag the *First_Name* row from the input table to the *Last_Name* row in the output table and drag the *Last_Name* row from the input table to the *First_Name* row in the output table.

Click **Apply** to save changes.

You will be prompted to propagate changes. Click **Yes** in the popup message.

Click **OK** to close the editor.

Select **tFileOutputDelimited** and click the **Component** tab to display the component view.

In the **Basic settings** view, set **tFileOutputDelimited** properties as needed.

Select **tChronometerStop** and click the **Component** tab to display the component view.

In the **Since options** panel of the **Basic settings** view, select **Since the beginning** option to measure the duration of the subjob as a whole.
• Select/clear the other check boxes as needed. In this scenario, we want to display the subjob duration on the console preceded by the component name.

• If needed, enter a text in the Caption field.

• Save your Job and press F6 to execute it.

Starting job tChronometerStop at 11:40 05/03/2010.
[statistics] connecting to socket on port 3399
[statistics] connected
[ tChronometerStop_1 ] 2seconds duration of the subjob
2968 milliseconds
[statistics] disconnected
Job tChronometerStop ended at 11:40 05/03/2010. [exit code=0]

Note:
You can measure the duration of the subjob the same way by placing tChronometerStop below tRowGenerator, and connecting the latter to tChronometerStop using an OnSubjobOk link.
tCloudStart

Starts instances on Amazon EC2 (Amazon Elastic Compute Cloud).

This component accesses the cloud provider to be used (Amazon EC2) and launches instances, which are virtual servers in that cloud. If an instance to be launched does not exist, tCloudStart creates it.

**tCloudStart Standard properties**

These properties are used to configure tCloudStart running in the Standard Job framework.

The Standard tCloudStart component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Access key and Secret key</th>
<th>Enter or paste the access key and the secret key required by Amazon to authenticate your requests to its web services. These access credentials are generated from the Security Credential tab of your Amazon account page. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud provider</td>
<td>Select the cloud provider to be used.</td>
</tr>
<tr>
<td>Image</td>
<td>Enter the name of the Amazon Machine Image (AMI) to be used to launch an instance. This AMI defines the basic configuration of that instance.</td>
</tr>
<tr>
<td>Region and Zone</td>
<td>Enter the region and the zone to be used as the geographic location where you want to launch an instance. The syntax used to express a location is predefined by Amazon, for example, us-east-1 representing the US East (Northern Virginia) region and us-east-1a representing one of the Availability Zones within that region. For further information about available regions for Amazon, see Amazon's documentation about regions and endpoints and as well Amazon's FAQ about region and Availability Zone.</td>
</tr>
<tr>
<td>Instance name</td>
<td>Enter the name of the instance to be launched. For example, you can enter Talend. Note that the upper letter will be converted to lower letter.</td>
</tr>
<tr>
<td>Instance count</td>
<td>Enter the number of instances to be launched. At runtime, the name specified in the Instance name field, for example Talend, will be used as the initial part of</td>
</tr>
</tbody>
</table>
### Instance type

Select the type of the instance(s) to be launched. Each type is predefined by Amazon and defines the performance of every instance you want to launch. This drop-down list presents the API name of each instance type. For further information, see Amazon’s documentation about instance types.

### Proceed with a Key pair

Select this check box to use Amazon Key Pair for your login to Amazon EC2. Once selecting it, a drop-down list appears to allow you to select:

- **Use an existing Key Pair** to enter the name of that Key Pair in the field next to the drop-down list. If required, Amazon will prompt you at runtime to find and use that Key Pair.

- **Create a Key Pair** to enter the name of the new Key Pair in the field next to the drop-down list and define the location where you want to store this Key Pair in the Advanced settings tab view.

### Security group

Add rows to this table and enter the names of the security groups to which you need to assign the instance(s) to be launched. The security groups set in this table must exist on your Amazon EC2.

A security group applies specific rules on inbound traffic to instances assigned to the group, such as the ports to be used. For further information about security groups, see Amazon’s documentation about security groups.

Note that an instance can be assigned to a group by setting its security group name or key pair name to `jclouds#$GROUP_NAME`, where `$GROUP_NAME` identifies the group to which the instance belongs. In this way, you can change the status of all instances or running instances in one group at the same time using the tCloudStop component.

### Advanced settings

#### Key Pair folder

Browse to, or enter the path to the folder you use to store the created Key Pair file.

This field appears when you select **Creating a Key Pair** in the Basic settings tab view.

#### Volumes

Add rows and define the volume(s) to be created for the instances to be launched in addition to the volumes predefined and allocated by the given Amazon EC2.

The parameters to be set in this table are the same parameters used by Amazon for describing a volume.

If you need to remove automatically an additional volume after terminating its related instance, select the check box in the Delete on termination column.
**tStatCatcher Statistics**

Select this check box to collect the log data at the component level.

---

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NODE_GROUP: the name of the instance. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NODES: the instances launched. This is an After variable and it returns an object.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

---

**Usage**

| Usage rule | This component works standalone to launch an instance on Amazon EC2. You can use this component to start the instance you need to deploy Jobs on. |

---

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tCloudStop

Changes the status of a launched instance on Amazon EC2 (Amazon Elastic Compute Cloud).
This component accesses the cloud provider to be used (Amazon EC2) and suspends, resumes or terminates given instance(s).

tCloudStop Standard properties

These properties are used to configure tCloudStop running in the Standard Job framework.
The Standard tCloudStop component belongs to the Cloud family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Access key and Secret key</th>
<th>Enter or paste the access key and the secret key required by Amazon to authenticate your requests to its web services. These access credentials are generated from the Security Credential view of your Amazon account page. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud provider</td>
<td>Select the cloud provider to be used.</td>
</tr>
<tr>
<td>Action</td>
<td>Select the action you need tCloudStop to take in order to change the status of a given instance. This action may be:</td>
</tr>
<tr>
<td></td>
<td>• Suspend</td>
</tr>
<tr>
<td></td>
<td>• Resume</td>
</tr>
<tr>
<td></td>
<td>• Terminate</td>
</tr>
<tr>
<td></td>
<td>Note that if you terminate an instance, this instance will be deleted, while a suspended instance can still be resumed.</td>
</tr>
<tr>
<td>Predicate</td>
<td>Select the instance(s) of which you need to change the status. The options are:</td>
</tr>
<tr>
<td></td>
<td>• Running instances: status of all the running instances will be changed.</td>
</tr>
<tr>
<td></td>
<td>• Instances in a specific group: status of the instances of a specific instance group will be changed. You need to enter the name of that group in the Group name field.</td>
</tr>
<tr>
<td></td>
<td>• Running instances in a specific group: status of the running instances of a specific instance group will be changed. You need to enter the name of that group in the Group name field.</td>
</tr>
</tbody>
</table>


- **Instance with predefined id**: status of a given instance will be changed. You need to enter the ID of that instance in the **Id** field. You can find this ID on your Amazon EC2.

An instance group is composed of the instances using the same instance name you have defined in the **Instance name** field of **TCloudStart**.

**Group name**

Enter the name of the group in which you want to change the status of given instances whose security group name or key pair name is set to `jclouds#$<group_name>` in the **TCloudStart** component, where `<$group_name>` identifies the group to which the instance belongs.

This field is available only when **Instances in a specific group** or **Running instances in a specific group** is selected from the **Predicate** list.

**Id**

Enter the ID of the instance of which you need to change the status. This field appears when you select **Instance with predefined id** from the **Predicate** list.

**Advanced settings**

- **tStatCatcher Statistics**
  
  Select this check box to collect the log data at the component level.

**Global Variables**

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

  A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

  To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

  For further information about variables, see **Talend Studio User Guide**.

**Usage**

- **Usage rule**

  This component works standalone to change the status of given instances on Amazon EC2. You can use this component to suspend, resume or terminate the instance(s) you have deployed Jobs on.

  This component often works alongside **TCloudStart** to change the status of the instances launched by the latter component.
Related scenarios

No scenario is available for the Standard version of this component yet.
tCombinedSQLAggregate

Provides a set of matrix based on values or calculations.

tCombinedSQLAggregate collects data values from one or more columns of a table for statistical purposes. This component has real-time capabilities since it runs the data transformation on the DBMS itself.

**tCombinedSQLAggregate Standard properties**

These properties are used to configure tCombinedSQLAggregate running in the Standard Job framework.

The Standard tCombinedSQLAggregate component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit schema | A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:
| | • **View schema**: choose this option to view the schema only.
| | • **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
| | • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata in the [Repository Content] window.
| | Click **Sync columns** to retrieve the schema from the previous component connected in the Job. |
| Built-in | **Built-in**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*. |
| Repository | **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Jobs. Related topic: see *Talend Studio User Guide*. |
| Group by | Define the aggregation sets, the values of which will be used for calculations. |
**Output Column**: Select the column label in the list offered according to the schema structure you defined. You can add as many output columns as you wish to make more precise aggregations.

**Input Column**: Select the input column label to match the output column’s expected content, in case the output label of the aggregation set needs to be different.

**Operations**
- **Output Column**: Select the destination field in the list.
- **Function**: Select any of the following operations to perform on data: `count`, `min`, `max`, `avg`, `sum`, `first`, `last`, `distinct` and `count (distinct)`.

**Input Column**: Select the input column from which you want to collect the values to be aggregated.

### Advanced settings

**tStatCatcher Statistics**
- Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

### Global Variables

**Global Variables**
- **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is a Flow variable and it returns an integer.
- **QUERY**: the query statement being processed. This is a Flow variable and it returns a string.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.

### Usage

**Usage rule**
- This component is an intermediary component. The use of the corresponding connection and commit components is recommended when using this...
Scenario: Filtering and aggregating table columns directly on the DBMS

The following scenario creates a Job that opens a connection to a MySQL database and:

- populates a database table with the input data,
- creates the output table for the filtered data,
- instantiates the schema from a database table in part (for column filtering),
- filters two columns in the same table to get only the data that meets two filtering conditions,
- collects data from the filtered column(s), grouped by specific value(s) and writes aggregated data in a target database table.
Adding and linking the components

Procedure

1. Drop the following components from the Palette onto the design workspace: tMysqlConnection, tFixedFlowInput, tMysqlOutput, tCreateTable, tCombinedSQLInput, tCombinedSQLFilter, tCombinedSQLAggregate, tCombinedSQLOutput, tMysqlCommit, tMysqlInput and tLogRow.

2. Connect tMysqlConnection to tFixedFlowInput using a Trigger > On Subjob Ok link.

3. Do the same to connect tFixedFlowInput to tCreateTable, tCreateTable to tCombinedSQLInput, tCombinedSQLInput to tMysqlCommit, and tMysqlCommit to tMysqlInput.

4. Connect tFixedFlowInput and tMysqlOutput using a Row > Main link.

5. Connect tCombinedSQLInput to tCombinedSQLFilter using a Row > Combine link.

6. Do the same to connect tCombinedSQLFilter to tCombinedSQLAggregate, and tCombinedSQLAggregate to tCombinedSQLOutput.

7. Connect tMysqlInput and tLogRow using a Row > Main link.
Configuring the components

The schema defined through `tCombinedSQLInput` can be different from that of the source table as you can just instantiate the desired columns of the source table. Therefore, `tCombinedSQLInput` also plays a role of column filtering.

In this scenario, the source database table has seven columns: `id`, `first_name`, `last_name`, `city`, `state`, `date_of_birth`, and `salary` while `tCombinedSQLInput` only instantiates four columns that are needed for the aggregation: `id`, `state`, `date_of_birth`, and `salary` from the source table.

Opening a MySQL connection

Procedure

1. Launch MySQL Workbench and start a local connection on port 3306.
2. Create a new schema and name it `test`.
3. Back in the design workspace, select `tMysqlConnection` and click the Component tab to define its basic settings.
4. In the Basic settings view, set the database connection details manually or select Repository from the Property Type list and select your DB connection if it has already been defined and stored in the Metadata area of the Repository tree view.

For more information on centralizing DB connection details in the Repository, see Talend Studio User Guide.
Populating the database table with input data

Procedure

1. In the design workspace, select tFixedFlowInput and click the Component tab to define its basic settings

   ![tFixedFlowInput_1](image)

   - **Basic settings**
     - **Number of rows**: 500

2. In the Basic settings view, in the Number of rows field, enter 500.

3. In this scenario, the source database table has seven columns: id, first_name, last_name, city, state, date_of_birth, and salary
   
   Click the [...] button next to Edit schema to define the following data structure.

   ![Schema of tFixedFlowInput_1](image)

4. Click the floppy disk icon to save the schema as a generic schema for later reuse.

5. In the Select folder window, select default and click OK.
6. Choose a name for your generic schema and click **Finish**.
7. Click **OK**.
8. The first column of the **Values** table automatically reflects the data structure you entered previously.
9. In the **Values** table, enter a value for each column.
10. In the design workspace, select **tMysqlOutput** and click the **Component** tab to define its basic settings.

   ![tMysqlOutput_1](image)

   The output schema will automatically be the same as the previous component, in this case **tFixedFlowInput**.

**Creating the target database table**

**Procedure**

1. In the design workspace, select **tCreateTable** and click the **Component** tab to define its basic settings.

   ![tCreateTable_1](image)

2. Click the [...] button next to **Edit schema** to define the following data structure.
The schema you enter at this step must reflect the different aggregation operations you want to perform on the input data.

**Extracting and filtering data**

**Procedure**

1. In the design workspace, select `tCombinedSQLInput` and click the Component tab to access the configuration panel.

2. Enter the source table name, in this case `employees` in the Table field.

3. In the Schema field, select Repository from the list and click the [...] button right to the empty field to load the schema you saved while configuring the settings for `tFixedFlowInput`.

4. In the Repository Content window, expand Generic schemas and select your schema.

5. Click the [...] button right to Edit schema.

6. Select View schema, and in the first column of the table, clear the check boxes for first_name, last_name and city.

**Filtering and aggregating the input data**

**Procedure**

1. In the design workspace, select `tCombinedSQLFilter` and click the Component tab to access the configuration panel.
2. Click the **Sync columns** button to retrieve the schema from the previous component, or configure the schema manually by selecting **Built-in** from the **Schema** list and clicking the [...] button next to **Edit schema**.

When you define the data structure for **tCombinedSQLFilter**, column names automatically appear in the **Input column** list in the **Conditions** table.

In this scenario, the **tCombinedSQLFilter** component instantiates four columns: *id*, *state*, *date_of_birth*, and *salary*.

3. In the **Conditions** table, set input parameters, operators and expected values in order to only extract the records that fulfill these criteria.

   Click two times on the [+] button under the **Conditions** table, and in **Input column**, select *state* and *date_of_birth* from the drop-down list.

   In this scenario, the **tCombinedSQLFilter** component filters the *state* and *date_of_birth* columns in the source table to extract the employees who were born after Oct. 19, 1960 and who live in the states **Utah**, **Ohio** and **Iowa**.

4. For the column *state*, select **IN** as operator from the drop-down list, and enter (*Utah*, *Ohio*, *Iowa*) as value.

5. For the column *date_of_birth*, select > as operator from the drop-down list, and enter (*1960-10-19*) as value.

6. Select **And** in the **Logical operator between conditions** list to apply the two conditions at the same time. You can also customize the conditions by selecting the **Use custom SQL** box and editing the conditions in the code box.

7. In the design workspace, select **tCombinedSQLAggregate** and click the **Component** tab to access the configuration panel.
8. Click on the [...] button next to **Edit schema** to enter the following configuration:

![Schema of tCombinedSQLAggregate_1](image)

The **tCombinedSQLAggregate** component instantiates four columns: *id*, *state*, *date_of_birth*, and *salary*, coming from the previous component.

9. The **Group by** table helps you define the data sets to be processed based on a defined column. In this example: *State*.

   In the **Group by** table, click the [*+] button to add one line.

10. In the **Output column** drop-down list, select *State*. This column will be used to hold the data filtered on *State*.

11. The **Operations** table helps you define the type of aggregation operations to be performed.

   The **Output column** list available depends on the schema you want to output (through the
tCombinedSQLAggregate component. In this scenario, we want to group employees based on the state they live in. Then we want to count the number of employees per state, calculate the average/lowest/highest salaries as well as the oldest/youngest employees for each state.

12. In the Operations table, click the [+] button to add a line and then click in the Output column list to select the output column that will hold the computed data.

13. In the Function field, select the relevant operation to be carried out.

Writing the output data into MySQL

Procedure

1. In the design workspace, select tCombinedSQLOutput and click the Component tab to access the configuration panel.

   ![tCombinedSQLOutput component configuration panel]

   2. On the Database type list, select the relevant database.

   3. On the Component list, select the relevant database connection component if more than one connection is used.

   4. In the Table field, enter the name of the target table which will store the results of the aggregation operations, empl_by_state in this case.

   The tCombinedSQLOutput component requires that an output table already exists in the database to work. That is why the empl_by_state table was created earlier in the scenario.

   In this example, the Schema field doesn’t need to be filled out as the database is not Oracle.

   5. Click the Sync columns button to retrieve the schema from the previous component.

In this scenario, tCombinedSQLOutput instantiates seven columns coming from the previous component in the Job design (tCombinedSQLAggregate): state, empl_count, avg_salary, min_salary, max_salary, oldest_empl and youngest_empl.

Committing the data into the database

Procedure

1. In the design workspace, select tCombinedSQLCommit and click the Component tab to access the configuration panel.

2. On the Component list, select the relevant database connection component if more than one connection is used.

3. Clear the check box Close Connection.
Retrieving the filtered and aggregated data

Procedure

1. In the design workspace, select tMysqlInput and click the Component tab to define its basic settings.

   ![tMysqlInput_1](image)

   - Use an existing connection
   - Schema: "empl_by_state"
   - Table Name: "empl_by_state"
   - Query Type: Built-In
   - Query: "select * from test.empl_by_state"

2. Select the check box Use an existing connection and choose tMysqlConnection_1 from the list.

3. Click on the [...] button next to Edit schema to enter the following schema:

   ![tMysqlInput_1](image)

   - Column: state
   - Db Column: state
   - Key: Yes
   - Type: String
   - DB Type: VARCHAR
   - N.: 30
   - Date: 0
   - Len.: 0
   - Pre.: 0

4. In the field Table Name, enter empl_by_state and in the Query field, enter select * from empl_by_state.

5. In the design workspace, select tLogRow and click the Component tab to define its basic settings.

   ![tLogRow_1](image)

   - Mode: Basic
   - Table (print values in cells of a table)
   - Print content with log4j
6. Click the **Sync columns** button to retrieve the schema from the previous component and select the **Table (print values in cells of a table)** mode.

### Saving and executing the Job

**Procedure**

1. Save your Job and press **F6** to execute it.
2. The **Run** tab opens, where you can observe the result of the Job execution.
3. The output data retrieved by the **tLogRow** is visible in a table.

### Results

Rows are inserted into a seven-column table `empl_by_state` in the database. The table shows, per defined state, the number of employees, the average salary, the lowest and highest salaries as well as the oldest and youngest employees.

```plaintext
```
tCombinedSQLFilter

Filters data by reorganizing, deleting or adding columns based on the source table and to filter the given data source using the filter conditions.

tCombinedSQLFilter allows you to alter the schema of a source table through column name mapping and to define a row filter on that table. Therefore, it can be used to filter columns and rows at the same time. This component has real-time capabilities since it runs the data filtering on the DBMS itself.

**tCombinedSQLFilter Standard properties**

These properties are used to configure tCombinedSQLFilter running in the Standard Job framework. The Standard tCombinedSQLFilter component belongs to the ELT family. The component in this framework is generally available.

**Basic settings**

| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  • **View schema**: choose this option to view the schema only.  
  • **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.  
  Click **Sync columns** to retrieve the schema from the previous component connected in the Job. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: You create and store the schema locally for this component only. Related topic: see  <strong>Talend Studio User Guide</strong>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Jobs. Related topic: see  <strong>Talend Studio User Guide</strong>.</td>
<td></td>
</tr>
<tr>
<td><strong>Logical operator between conditions</strong></td>
<td>Select the logical operator between the filter conditions defined in the <strong>Conditions</strong> panel.</td>
</tr>
</tbody>
</table>
Two operators are available: Or, And.

**Conditions**
Select the type of WHERE clause along with the values and the columns to use for row filtering.

**Input Column:** Select the column to filter in the list.

**Operator:** Select the type of the WHERE clause: =, <>, >, <, >=, <=, LIKE, IN, NOT IN, and EXIST IN.

**Values:** Type in the values to be used in the WHERE clause.

**Negate:** Select this check box to enable the condition that is opposite to the current setting.

**Use custom SQL**
Customize a WHERE clause by selecting this check box and editing in the **SQL Condition** field.

**Advanced settings**

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

**Global Variables**

**Global Variables**

- **NB_LINE:** the number of rows read by an input component or transferred to an output component. This is a Flow variable and it returns an integer.
- **QUERY:** the query statement being processed. This is a Flow variable and it returns a string.
- **ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.

**Usage**

**Usage rule**
This component is an intermediary component. The use of the corresponding connection and commit components is recommended when using this component to allow a unique connection to be open and then closed during the Job execution.
Related Scenario

For a related scenario, see Scenario: Filtering and aggregating table columns directly on the DBMS on page 467.
tCombinedSQLInput

Extracts fields from a database table based on its schema definition.

Then it passes on the field list to the next component via a Combine row link. The schema of tCombinedSQLInput can be different from that of the source database table but must correspond to it in terms of the column order.

tCombinedSQLInput extracts fields from a database table based on its schema. This component also has column filtering capabilities since its schema can be different from that of the database table.

tCombinedSQLInput Standard properties

These properties are used to configure tCombinedSQLInput running in the Standard Job framework. The Standard tCombinedSQLInput component belongs to the ELT family. The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Table</th>
<th>Name of the source database table.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>Name of the source table’s schema. This field has to be filled if the database is Oracle.</td>
</tr>
</tbody>
</table>
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  • **View schema**: choose this option to view the schema only.  
  • **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
  • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
| Built-in | You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide. |
| Repository | You have already created the schema and stored it in the Repository. You can reuse it in various projects and Jobs. Related topic: see Talend Studio User Guide. |
Add additional columns

This option allows you to call SQL functions to perform actions on columns, provided that these are not insert, update or delete actions, or actions that require pre-processing.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type in the name of the schema column to be altered.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL expression</td>
<td>Type in the SQL statement to be executed in order to alter the data in the corresponding column.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

Global Variables

| Global Variables | NB_LINE: the number of rows read by an input component or transferred to an output component. This is a Flow variable and it returns an integer.  
QUERY: the query statement being processed. This is a Flow variable and it returns a string.  
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component is an intermediary component. The use of the corresponding connection and commit components is recommended when using this component to allow a unique connection to be open and then closed during the Job execution. |

Related scenario

For a related scenario, see Scenario: Filtering and aggregating table columns directly on the DBMS on page 467.
**tCombinedSQLOutput**

Inserts records from the incoming flow to an existing database table.

**tCombinedSQLOutput Standard properties**

These properties are used to configure tCombinedSQLOutput running in the Standard Job framework. The Standard tCombinedSQLOutput component belongs to the ELT family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Select the database type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the relevant DB connection component in the list if more than one connection is used for the current Job.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the target database table.</td>
</tr>
<tr>
<td>Schema</td>
<td>Name of the target database table’s schema. This field has to be filled if the database is Oracle.</td>
</tr>
</tbody>
</table>

**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

- **Built-in**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Jobs. Related topic: see *Talend Studio User Guide*.
<table>
<thead>
<tr>
<th>Action on data</th>
<th>Select <strong>INSERT</strong> from the list to insert the records from the incoming flow to the target database table.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advanced settings</strong></td>
<td></td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
<tr>
<td><strong>Global Variables</strong></td>
<td></td>
</tr>
</tbody>
</table>
| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is a Flow variable and it returns an integer.  
**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see **Talend Studio User Guide**. |
| **Usage** | | |
| Usage rule | This component is an intermediary component. The use of the corresponding connection and commit components is recommended when using this component to allow a unique connection to be open and then closed during the Job execution. |
| **Related scenario** | | |
| For a related scenario, see Scenario: Filtering and aggregating table columns directly on the DBMS on page 467. | |
tContextDump

Copies the context setup of the current Job to a flat file, a database table, etc., which can then be used by tContextLoad.

Together with tContextLoad, this component makes it simple to apply the context setup of one Job to another.

tContextDump dumps the context setup of the current Job to the subsequent component.

**tContextDump Standard properties**

These properties are used to configure tContextDump running in the Standard Job framework.

The Standard tContextDump component belongs to the Misc family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description, it defines the fields that will be processed and passed on to the next component.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note: The schema of tContextDump is read only and made up of two columns, Key and Value, corresponding to the parameter name and the parameter value of the Job context.</td>
</tr>
</tbody>
</table>

| Hide Password | Select this check box to hide the value of context parameter password, namely displaying the value of context parameters whose Type is Password as "." |

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
Usage

| Usage rule | As a start component, **tContextDump** dumps the context setup of the current Job to a file, a database table, etc. |

Related scenarios

No scenario is available for the Standard version of this component yet.
tContextLoad

Loads a context from a flow.

This component performs also two controls. It warns when the parameters defined in the incoming flow are not defined in the context, and the other way around, it also warns when a context value is not initialized in the incoming flow. But note that this does not block the processing.

tContextLoad modifies dynamically the values of the active context.

**tContextLoad Standard properties**

These properties are used to configure tContextLoad running in the Standard Job framework.

The Standard tContextLoad component belongs to the Misc family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Schema and Edit schema</strong></th>
<th>A schema is a row description, it defines the fields that will be processed and passed on to the next component. In tContextLoad, the schema must be made of two columns, including the parameter name and the parameter value to be loaded.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>If a variable loaded, but not in the context</strong></td>
<td>If a variable is loaded but does not appear in the context, select how the notification must be displayed. In the shape of an Error, a warning or an information (info).</td>
</tr>
<tr>
<td><strong>If a variable in the context, but not loaded</strong></td>
<td>If a variable appears in the context but is not loaded, select how the notification must be displayed. In the shape of an Error, a warning or an information (info).</td>
</tr>
<tr>
<td><strong>Print operations</strong></td>
<td>Select this check box to display the context parameters set in the Run view.</td>
</tr>
<tr>
<td><strong>Disable errors</strong></td>
<td>Select this check box to prevent the error from displaying.</td>
</tr>
<tr>
<td><strong>Disable warnings</strong></td>
<td>Select this check box to prevent the warning from displaying.</td>
</tr>
<tr>
<td><strong>Disable infos</strong></td>
<td>Select this check box to prevent the information from displaying.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows.</td>
</tr>
</tbody>
</table>
### Advanced settings

| tStat Catcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

### Global Variables

| Global Variables | **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
| | **KEY_NOT_INCONTEXT**: the variables are loaded but do not appear in the context. This is an After variable and it returns a string.  
| | **KEY_NOT_LOADED**: the variables not loaded but appear in the context. This is an After variable and it returns a string.  
| | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
| | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
| | To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
| | For further information about variables, see Talend Studio User Guide. |

### Usage

| Usage rule | This component relies on the data flow to load the context values to be used, therefore it requires a preceding input component and thus cannot be a start component. |

| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to turn on or off the **Print operations** option dynamically at runtime.  
| | When a dynamic parameter is defined, the corresponding **Print operations** option in the **Basic settings** view becomes unusable.  
| | For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |
Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters

The Job in this scenario is made of two subjobs. The first subjob aims at dynamically loading the context parameters from two text files, and the second subjob uses the loaded context parameters to connect to two different databases and to display the content of an existing database table of each of them. With the context settings in the Job, we can decide which database to connect to and choose whether to display the set context parameters on the console dynamically at runtime.

Dropping and linking the components

Procedure

1. Drop a `tFileInputDelimited` component and a `tContextLoad` component from the Palette onto the design workspace, and link them using a Row > Main connection to form the first subjob.
2. Drop a `tMysqlInput` component and a `tLogRow` component onto the design workspace, and link them using a Row > Main connection to form the second subjob.
3. Link the two subjobs using a Trigger > On Subjob Ok connection.

Preparing the contexts and context variables

Procedure

1. Create two delimited files corresponding to the two contexts in this scenario, namely two databases we will access, and name them `test_connection.txt` and `prod_connection.txt`, which contain the database connection details for testing and actual production purposes respectively. Each file is made of two columns, containing the parameter names and the corresponding values respectively. Below is an example:

```
host=localhost
port;3306
database;test
username;root
```
2. Select the **Contexts** view of the Job, and click the [+](#) button at the bottom of the view to add seven rows in the table to define the following parameters:
   - **host**, String type
   - **port**, String type
   - **database**, String type
   - **username**, String type
   - **password**, Password type
   - **filename**, File type
   - **printOperations**, Boolean type

   Note that the **host**, **port**, **database**, **username** and **password** parameters correspond to the parameter names in the delimited files and are used to set up the desired database connection, the **filename** parameter is used to define the delimited file to read at Job execution, the **printOperations** parameter is used to decide whether to print the context parameters set by the [tContextLoad](#) component on the console.

3. Click the **Contexts** tab and click the [+](#) button at the upper right corner of the panel to open the [Configure Contexts](#) dialog box.
4. Select the default context, click the **Edit** button and rename the context to **Test**.
5. Click **New** to add a new context named **Production**. Then click **OK** to close the dialog box.

6. Back in the **Contexts** tab view, define the value of the **filename** variable under each context by clicking in the respective **Value** field and browse to the corresponding delimited file.
7. Select the **Prompt** check box next to the **Value** field of the **filename** variable for both contexts to show the **Prompt** fields and enter the prompt message to be displayed at the execution time.
8. For the **printOperations** variable, click in the **Value** field under the **Production** context and select **false** from the list; click in the **Value** field under the **Test** context and select **true** from the list. Then
select the **Prompt** check box under both contexts and enter the prompt message to be displayed at the execution time.

### Configuring the components

**Procedure**

1. In the **tFileInputDelimited** component **Basic settings** panel, fill the **File name/Stream** field with the relevant context variable we just defined: `context.filename`.

2. Define the file schema manually (Built-in). It contains two columns defined as: **key** and **value**.

3. Accept the defined schema to be propagated to the next component (**tContextLoad**).

4. In the **Dynamic settings** view of the **tContextLoad** component, click the `[*]` button to add a row in the table, and fill the **Code** field with `context.printOperations` to use context variable `printOperations` we just defined. Note that the **Print operations** check box in the **Basic settings** view now becomes highlighted and unusable.
5. Then double-click to open the **tMysqlInput** component **Basic settings** view.

6. Fill the **Host**, **Port**, **Database**, **Username**, and **Password** fields with the relevant variables stored in the delimited files and defined in the **Contexts** tab view: `context.host`, `context.port`, `context.database`, `context.username`, and `context.password` respectively in this example, and fill the **Table Name** field with the actual database table name to read data from, `customers` for both databases in this example.

7. Then fill in the **Schema** information. If you stored the schema in the **Repository Metadata**, then you can retrieve it by selecting **Repository** and the relevant entry in the list.

   In this example, the schema of both database tables is made of four columns: `id` (INT, 2 characters long), `firstName` (VARCHAR, 15 characters long), `lastName` (VARCHAR, 15 characters long), and `city` (VARCHAR, 15 characters long).

8. In the **Query** field, type in the SQL query to be executed on the DB table specified. In this example, simply click **Guess Query** to retrieve all the columns of the table, which will be displayed on the **Run** tab, through the **tLogRow** component.

9. In the **Basic settings** view of the **tLogRow** component, select the **Table** option to display data records in the form of a table.
Executing the Job

Procedure

1. Press Ctrl+S to save the Job, and press F6 to run the Job using the default context, which is Test in this use case.

A dialog box appears to prompt you to specify the delimited file to read and decide whether to display the set context parameters on the console.

You can specify a file other than the default one if needed, and clear the Show loaded variables check box if you do not want to see the set context variables on the console. To run the Job using the default settings, click OK.

The context parameters and content of the database table in the Test context are all displayed on the Run console.
2. Now select the Production context and press **F6** to launch the Job again. When the prompt dialog box appears, simply click **OK** to run the Job using the default settings.

The content of the database table in the Production context is displayed on the Run console. Because the `printOperations` variable is set to `false`, the set context parameters are not displayed on the console this time.
**tConvertType**

Converts one Talend java type to another automatically, and thus avoid compiling errors.
tConvertType allows specific conversions at runtime from one Talend java type to another.

**tConvertType Standard properties**

These properties are used to configure tConvertType running in the Standard Job framework.
The Standard tConvertType component belongs to the Processing family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
</tbody>
</table>

| Built-In | You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide. |
| Repository | You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. |
| Auto Cast | This check box is selected by default. It performs an automatic java type conversion. |
| Manual Cast | This mode is not visible if the Auto Cast check box is selected. It allows you to precise manually the columns where a java type conversion is needed. |
| Set empty values to Null before converting | This check box is selected to set the empty values of String or Object type to null for the input data. |
Die on error

Note: Not available for Map/Reduce Jobs.

This check box is selected to kill the Job when an error occurs.

Advanced settings

tStatCatcher Statistics

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

Global Variables

Global Variables

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

This component cannot be used as a start component as it requires an input flow to operate.

Scenario: Converting java types

This Java scenario describes a four-component Job where the tConvertType component is used to convert Java types in three columns, and a tMap is used to adapt the schema and have as an output the first of the three columns and the sum of the two others after conversion.

Note:

In this scenario, the input schemas for the input delimited file are stored in the repository, you can simply drag and drop the relevant file node from Repository - Metadata - File delimited onto the design workspace to automatically retrieve the tFileInputDelimited component’s setting. For more information, see Talend Studio User Guide.
Dropping the components

Procedure
1. Drop the following components from the Palette onto the design workspace: tConvertType, tMap, and tLogRow.
2. In the Repository tree view, expand Metadata and from File delimited drag the relevant node, JavaTypes in this scenario, to the design workspace.

   The [Components] dialog box displays.
3. From the component list, select tFileInputDelimited and click Ok.

   A tFileInputComponent called Java types displays in the design workspace.
4. Connect the components using Row > Main links.

Configuring the components

Procedure
1. Double-click tFileInputDelimited to enter its Basic settings view.
2. Set Property Type to Repository since the file details are stored in the repository. The fields to follow are pre-defined using the fetched data.

   The input file used in this scenario is called input. It is a text file that holds string, integer, and float java types.

   Fill in all other fields as needed. For more information, see tFileInputDelimited on page 1010.

   In this scenario, the header and the footer are not set and there is no limit for the number of processed rows.
3. Click **Edit schema** to describe the data structure of this input file. In this scenario, the schema is made of three columns, *StringToInteger*, *IntegerField*, and *FloatToInteger*.

4. Click **Ok** to close the dialog box.

5. Double-click **tConvertType** to enter its **Basic settings** view.

6. Set **Schema Type** to **Built in**, and click **Sync columns** to automatically retrieve the columns from the **tFileInputDelimited** component.

7. Click **Edit schema** to describe manually the data structure of this processing component.

8. Double-click **tMap** to open the Map editor.

   The Map editor displays the input metadata of the **tFileInputDelimited** component.
9. In the **Schema editor** panel of the Map editor, click the plus button of the output table to add two rows and name them to *StringToInteger* and *Sum*.

10. In the Map editor, drag the *StringToInteger* row from the input table to the *StringToInteger* row in the output table.

11. In the Map editor, drag each of the *IntegerField* and the *FloatToInteger* rows from the input table to the *Sum* row in the output table and click **OK** to close the Map editor.

12. In the design workspace, select **tLogRow** and click the **Component** tab to define its basic settings. For more information, see **tLogRow** on page 1988.
Executing the Job

Procedure

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to execute it.

```
Starting job tConvertType at 10:03 19/11/2006.
<table>
<thead>
<tr>
<th>tLogRow_1</th>
<th>StringToInteger</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sun</td>
<td>579</td>
</tr>
</tbody>
</table>
```

`Job tConvertType ended at 10:03 19/11/2006. `

The string type data is converted into an integer type and displayed in the `StringToInteger` column on the console. The float type data is converted into an integer and added to the `IntegerField` value to give the addition result in the `Sum` column on the console.
tCosmosDBBulkLoad

Imports data files in different formats (CSV, TSV or JSON) into the specified Cosmos database so that the data can be further processed.

tCosmosDBBulkLoad Standard properties

These properties are used to configure tCosmosDBBulkLoad running in the Standard Job framework. The Standard tCosmosDBBulkLoad component belongs to the Cloud and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- View schema: choose this option to view the schema only.
- Change to built-in property: choose this option to change the schema to Built-in for local changes.
- Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |

| MongoDB directory | Fill in this field with the MongoDB home directory. |

| Use replica set address or multiple query routers | Select this check box to show the Server addresses table.

In the Server addresses table, define the sharded MongoDB databases or the MongoDB replica sets you want to connect to. |

| Server and Port | Enter the IP address and listening port of the database server.

Available when the Use replica set address or multiple query routers check box is not selected. |

| Database | Enter the name of the MongoDB database to be connected to. |

<p>| Collection | Type in the name of the collection to import data to. |</p>
<table>
<thead>
<tr>
<th>Drop collection if exist</th>
<th>Select this check box to remove the collection if it already exists.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication mechanism</td>
<td>Among the mechanisms listed on the Authentication mechanism drop-down list, the NEGOTIATE one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using. For details about the other mechanisms in this list, see MongoDB Authentication from the MongoDB documentation.</td>
</tr>
<tr>
<td>Set Authentication database</td>
<td>If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name of this Authentication database in the Authentication database field that is displayed. For further information about the MongoDB Authentication database, see User Authentication database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. Available when the Required authentication check box is selected. If the security system you have selected from the Authentication mechanism drop-down list is Kerberos, you need to enter the User principal, the Realm and the KDC server fields instead of the Username and the Password fields.</td>
</tr>
<tr>
<td>Data file</td>
<td>Type in the full path of the file from which the data will be imported or click the [...] button to browse to the desired data file. Make sure that the data file is in standard format. For example, the fields in CSV files should be separated with commas.</td>
</tr>
<tr>
<td>File type</td>
<td>Select the proper file type from the list. CSV, TSV and JSON are supported.</td>
</tr>
<tr>
<td>The JSON file starts with an array</td>
<td>Select this check box to allow tCosmosDBBulkload to read the JSON files starting with an array. This check box appears when the File type you have selected is JSON.</td>
</tr>
<tr>
<td>Action on data</td>
<td>Select the action that you want to perform on the data. • Insert: Insert the data into the database. Note that when inserting data from CSV or TSV files into the MongoDB database, you need to specify</td>
</tr>
</tbody>
</table>
- **First line is header**: Check box to use the first line in CSV or TSV files as a header.
  - This check box is available only when you select CSV or TSV from the File type list.

- **Ignore blanks**: Check box to ignore the empty fields in CSV or TSV files.
  - This check box is available only when you select CSV or TSV from the File type list.

- **Print log**: Check box to print logs.

**Advanced settings**

- **Additional arguments**: Complete this table to use the additional arguments as required.
  - For example, you can use the argument `--jsonArray` to accept the import of data expressed with multiple MongoDB documents within a single JSON array. For more information about the additional arguments, go to [http://docs.mongodb.org/manual/reference/program/mongoimport/](http://docs.mongodb.org/manual/reference/program/mongoimport/) and read the description of options.

- **tStatCatcher Statistics**: Check box to collect the log data at a component level.

**Usage**

- **Usage rule**: This component can be used together with the tCosmosDBInput component to verify if the data is imported as expected.

- **Limitation**: The MongoDB client tool needs to be installed on the machine where Jobs using this component are executed.
tCosmosDBConnection

Creates a connection to a CosmosDB database and reuse that connection in other components.

**tCosmosDBConnection Standard properties**

These properties are used to configure tCosmosDBConnection running in the Standard Job framework.

The Standard tCosmosDBConnection component belongs to the Cloud and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API</strong></td>
<td>Select the database API to be used. Then the corresponding parameters to be defined are displayed in the Component view. In the current version of this component, only the MongoDB API is supported. For this reason, MongoDB database is often mentioned in the documentation of the CosmosDB components.</td>
</tr>
<tr>
<td><strong>Use replica set address or multiple query routers</strong></td>
<td>Select this check box to show the Server addresses table. In the Server addresses table, define the sharded MongoDB databases or the MongoDB replica sets you want to connect to.</td>
</tr>
<tr>
<td><strong>Server and Port</strong></td>
<td>Enter the IP address and listening port of the database server. Available when the Use replica set address or multiple query routers check box is not selected.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Enter the name of the MongoDB database to be connected to.</td>
</tr>
<tr>
<td><strong>Authentication mechanism</strong></td>
<td>Among the mechanisms listed on the Authentication mechanism drop-down list, the NEGOTIATE one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using. For details about the other mechanisms in this list, see MongoDB Authentication from the MongoDB documentation.</td>
</tr>
<tr>
<td><strong>Set Authentication database</strong></td>
<td>If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name of this Authentication database in the Authentication database field that is displayed.</td>
</tr>
</tbody>
</table>
For further information about the MongoDB Authentication database, see User Authentication database.

**Username and Password**

DB user authentication data.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

Available when the Use authentication check box is selected.

If the security system you have selected from the Authentication mechanism drop-down list is Kerberos, you need to enter the User principal, the Realm and the KDC server fields instead of the Username and the Password fields.

---

**Advanced settings**

**tStatCatcher Statistics**

Select this check box to collect the log data at a component level.

**No query timeout**

Select this check box to prevent MongoDB servers from stopping idle cursors at the end of 10-minute inactivity of these cursors. In this situation, an idle cursor will stay open until either the results of this cursor are exhausted or you manually close it using the cursor.close() method.

A cursor for MongoDB is a pointer to the result set of a query. By default, that is to say, with this check box being clear, a MongoDB server automatically stops idle cursors after a given inactivity period to avoid excess memory use. For further information about MongoDB cursors, see https://docs.mongodb.org/manual/core/cursors/.

---

**Usage**

**Usage rule**

This component is generally used with other CosmosDB components, particularly tCosmosClose.
tCosmosDBInput

Retrieves certain documents from a Cosmos database collection by supplying a query document containing the fields the desired documents should match.

**tCosmosDBInput Standard properties**

These properties are used to configure tCosmosDBInput running in the Standard Job framework. The Standard tCosmosDBInput component belongs to the Cloud and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>API</strong></td>
<td>Select the database API to be used. Then the corresponding parameters to be defined are displayed in the Component view. In the current version of this component, only the MongoDB API is supported. For this reason, MongoDB database is often mentioned in the documentation of the CosmosDB components.</td>
</tr>
<tr>
<td><strong>Use replica set address or multiple query routers</strong></td>
<td>Select this check box to show the Server addresses table. In the Server addresses table, define the sharded MongoDB databases or the MongoDB replica sets you want to connect to.</td>
</tr>
<tr>
<td><strong>Server and Port</strong></td>
<td>Enter the IP address and listening port of the database server. Available when the Use replica set address or multiple query routers check box is not selected.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Enter the name of the MongoDB database to be connected to.</td>
</tr>
<tr>
<td><strong>Set read preference</strong></td>
<td>Select this check box and from the Read preference drop-down list that is displayed, select the member to which you need to direct the read operations. If you leave this check box clear, the Job uses the default Read preference, that is to say, uses the primary member in a replica set. For further information, see MongoDB’s documentation about Replication and its Read preferences.</td>
</tr>
<tr>
<td><strong>Authentication mechanism</strong></td>
<td>Among the mechanisms listed on the Authentication mechanism drop-down list, the NEGOTIATE one is</td>
</tr>
</tbody>
</table>
recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using.

For details about the other mechanisms in this list, see MongoDB Authentication from the MongoDB documentation.

**Set Authentication database**

If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name of this Authentication database in the Authentication database field that is displayed.

For further information about the MongoDB Authentication database, see User Authentication database.

**Username** and **Password**

DB user authentication data.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

Available when the Use authentication check box is selected.

If the security system you have selected from the Authentication mechanism drop-down list is Kerberos, you need to enter the User principal, the Realm and the KDC server fields instead of the Username and the Password fields.

**Collection**

Name of the collection in the database.

**Schema** and **Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

If a column in the database is a JSON document and you need to read the entire document, put an asterisk (*) in the DB column column, without quotation marks around.
### Query

Specify the query condition. This field is available only when you have selected **Find query** from the **Query type** drop-down list.

For example, type in "id:4" to retrieve the record whose id is 4 from the collection specified in the **Collection** field.

Different from the query statements required in the MongoDB client software, the query here refers to the contents inside `find()`, such as the query here `{id:4}` versus the MongoDB client query `db.blog.find({id:4})`.

### Mapping

Each column of the schema defined for this component represents a field of the documents to be read. In this table, you need to specify the parent nodes of these fields, if any.

For example, in the document reading as follows

```json
{
    _id: ObjectId("5099803df3f4948bd2f98391"),
    person: { first: "Joe", last: "Walker" }
}
```

The `first` and the `last` fields have `person` as their parent node but the `_id` field does not have any parent node. So once completed, this **Mapping** table should read as follows:

<table>
<thead>
<tr>
<th>Column</th>
<th>Parent node path</th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td></td>
</tr>
<tr>
<td>first</td>
<td>&quot;person&quot;</td>
</tr>
<tr>
<td>last</td>
<td>&quot;person&quot;</td>
</tr>
</tbody>
</table>

### Sort by

Specify the column and choose the order for the sort operation.

This field is available only when you have selected **Find query** from the **Query type** drop-down list.

### Limit

Type in the maximum number of records to be retrieved.

This field is available only when you have selected **Find query** from the **Query type** drop-down list.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to collect the log data at the component level.

**No query timeout**

Select this check box to prevent MongoDB servers from stopping idle cursors at the end of 10-minute inactivity of these cursors. In this situation, an idle cursor will stay open until either the results of this cursor are exhausted or you manually close it using the `cursor.close()` method.
A cursor for MongoDB is a pointer to the result set of a query. By default, that is to say, with this check box being clear, a MongoDB server automatically stops idle cursors after a given inactivity period to avoid excess memory use. For further information about MongoDB cursors, see [https://docs.mongodb.org/manual/core/cursors/](https://docs.mongodb.org/manual/core/cursors/).

**Enable external sort**

Since the aggregation pipeline stages have a maximum memory use limit (100 megabytes) and a stage exceeding this limit will produce errors, when handling large datasets, select this check box to avoid aggregation stages exceeding this limit.

For further information about this external sort, see [Large sort operation with external sort.](#)

**Usage**

**Usage rule**

As a start component, `tCosmosDBInput` allows you to retrieve records from a collection in the Cosmos database and transfer them to the following component for display or storage.
tCosmosDBOutput

Inserts, updates, upserts or deletes documents in a Cosmos database collection based on the incoming flow from the preceding component in the Job.

**tCosmosDBOutput Standard properties**

These properties are used to configure tCosmosDBOutput running in the Standard Job framework.

The Standard tCosmosDBOutput component belongs to the Cloud and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Use existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API</strong></td>
<td>Select the database API to be used. Then the corresponding parameters to be defined are displayed in the Component view.</td>
</tr>
<tr>
<td></td>
<td>In the current version of this component, only the MongoDB API is supported. For this reason, MongoDB database is often mentioned in the documentation of the CosmosDB components.</td>
</tr>
<tr>
<td><strong>Use replica set address or multiple query routers</strong></td>
<td>Select this check box to show the Server addresses table.</td>
</tr>
<tr>
<td></td>
<td>In the Server addresses table, define the sharded MongoDB databases or the MongoDB replica sets you want to connect to.</td>
</tr>
<tr>
<td><strong>Server and Port</strong></td>
<td>Enter the IP address and listening port of the database server.</td>
</tr>
<tr>
<td></td>
<td>Available when the Use replica set address or multiple query routers check box is not selected.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Enter the name of the MongoDB database to be connected to.</td>
</tr>
<tr>
<td><strong>Set write concern</strong></td>
<td>Select this check box to set the level of acknowledgement requested from for write operations. Then you need to select the level of this operation.</td>
</tr>
<tr>
<td></td>
<td>For further information, see the related MongoDB documentation on <a href="http://docs.mongodb.org/manual/core/write-concern/">http://docs.mongodb.org/manual/core/write-concern/</a>.</td>
</tr>
<tr>
<td><strong>Bulk write</strong></td>
<td>Select this check box to insert, update or remove data in bulk. Note this feature is available only when the version of MongoDB you are using is 2.6+.</td>
</tr>
</tbody>
</table>
Then you need to select **Ordered** or **Unordered** to define how the MongoDB database processes the data sent by the Studio.

- If you select **Ordered**, MongoDB processes the queries sequentially.
- If you select **Unordered**, MongoDB optimizes the bulk write operations without keeping the order in which the individual operations were inserted in the bulk write.

In the **Bulk write size** field, enter the size of each query group to be processed by MongoDB. In the documentation of MongoDB, some restrictions and expected behaviors as to this size are explained. You can find the details on [http://docs.mongodb.org/manual/core/bulk-write-operations/](http://docs.mongodb.org/manual/core/bulk-write-operations/).

<table>
<thead>
<tr>
<th>Authentication mechanism</th>
<th>Authentication mechanism drop-down list, the <strong>NEGOTIATE</strong> one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using. For details about the other mechanisms in this list, see <strong>MongoDB Authentication</strong> from the MongoDB documentation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Authentication database</td>
<td>If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name of this Authentication database in the <strong>Authentication database</strong> field that is displayed. For further information about the MongoDB Authentication database, see <strong>User Authentication database</strong>.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings. Available when the <strong>Use authentication</strong> check box is selected. If the security system you have selected from the <strong>Authentication mechanism</strong> drop-down list is Kerberos, you need to enter the <strong>User principal</strong>, the <strong>Realm</strong> and the <strong>KDC server</strong> fields instead of the <strong>Username</strong> and the <strong>Password</strong> fields.</td>
</tr>
<tr>
<td>Collection</td>
<td>Name of the collection in the database.</td>
</tr>
<tr>
<td>Drop collection if exist</td>
<td>Select this check box to drop the collection if it already exists.</td>
</tr>
</tbody>
</table>
| Action on data | The following operations are available:  
  - **Insert**: insert documents. |
• **Set**: modifies the existing fields of an existing document and appends a field if it does not exist in this document.

If you need to apply this action on all the documents in the collection to be used, select the **Update all document** check box that is displayed; otherwise, only the first document is updated.

• **Update**: replaces the existing documents with the incoming data but keeps the technical ID of these documents.

• **Upset**: inserts a document if it does not exist otherwise it applies the same rules as **Update**.

• **Upset with set**: inserts a document if it does not exist otherwise it applies the same rules as **Set**

If you need to apply this action on all the documents in the collection to be used, select the **Update all document** check box that is displayed; otherwise, only the first document is updated.

• **Delete**: delete documents.

### Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word *line* when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

• **View schema**: choose this option to view the schema only.

• **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

| **Built-In** | You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*. |
| **Repository** | You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*. |

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.
You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

### Mapping

Each column of the schema defined for this component represents a field of the documents to be read. In this table, you need to specify the parent nodes of these fields, if any.

For example, in the document reading as follows

```json
{
  _id: ObjectId("5099803df3f4948bd2f98391"),
  person: { first: "Joe", last: "Walker" }
}
```

The `first` and the `last` fields have `person` as their parent node but the `_id` field does not have any parent node. So once completed, this **Mapping** table should read as follows:

<table>
<thead>
<tr>
<th>Column</th>
<th>Parent node path</th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td></td>
</tr>
<tr>
<td>first</td>
<td>&quot;person&quot;</td>
</tr>
<tr>
<td>last</td>
<td>&quot;person&quot;</td>
</tr>
</tbody>
</table>

The `first` and the `last` fields have `person` as their parent node but the `_id` field does not have any parent node. So once completed, this **Mapping** table should read as follows:

<table>
<thead>
<tr>
<th>Column</th>
<th>Parent node path</th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td></td>
</tr>
<tr>
<td>first</td>
<td>&quot;person&quot;</td>
</tr>
<tr>
<td>last</td>
<td>&quot;person&quot;</td>
</tr>
</tbody>
</table>

Not available when the **Generate JSON Document** check box is selected in **Advanced settings**.

### Die on error

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

### Advanced settings

#### Generate JSON Document

Select this check box for JSON configuration:

- **Configure JSON Tree**: click the [...] button to open the interface for JSON tree configuration. For more information, see Configuring a JSON Tree on page 4040.
- **Group by**: click the [+ ] button to add lines and choose the input columns for grouping the records.
- **Remove root node**: select this check box to remove the root node.
- **Data node** and **Query node** (available for update and upsert actions): type in the name of data node and query node configured on the JSON tree.

These nodes are mandatory for update and upsert actions. They are intended to enable the update and upsert actions though will not be stored in the database.

#### No query timeout

Select this check box to prevent MongoDB servers from stopping idle cursors at the end of 10-minute inactivity of these cursors. In this situation, an idle cursor will stay open until either the results of this cursor are exhausted or you manually close it using the `cursor.close()` method.
A cursor for MongoDB is a pointer to the result set of a query. By default, that is to say, with this check box being clear, a MongoDB server automatically stops idle cursors after a given inactivity period to avoid excess memory use. For further information about MongoDB cursors, see [https://docs.mongodb.org/manual/core/cursors/](https://docs.mongodb.org/manual/core/cursors/).

| tStatCatcher Statistics | Select this check box to collect the log data at the component level. |

**Usage**

| Usage rule | tCosmosDBOutput executes the action defined on the collection in the database based on the flow incoming from the preceding component in the Job. |

| Limitation | • The "multi" parameter, which allows to update multiple documents at a time, is not supported. Therefore, if two documents have the same key, the first is always updated, but the second never will.  
• For the update operation, the key cannot be a JSON array. |
tCosmosDBRow

Executes the commands of the Cosmos database.

**tCosmosDBRow Standard properties**

These properties are used to configure tCosmosDBRow running in the Standard Job framework.

The Standard tCosmosDBRow component belongs to the Cloud and the Databases families.

The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>API</td>
<td>Select the database API to be used. Then the corresponding parameters to be defined are displayed in the <strong>Component</strong> view.</td>
</tr>
<tr>
<td>Use replica set address or multiple query routers</td>
<td>Select this check box to show the <strong>Server addresses</strong> table.</td>
</tr>
<tr>
<td>Use replica set address or multiple query routers</td>
<td>In the <strong>Server addresses</strong> table, define the sharded MongoDB databases or the MongoDB replica sets you want to connect to.</td>
</tr>
<tr>
<td>Server and Port</td>
<td>Enter the IP address and listening port of the database server.</td>
</tr>
<tr>
<td>Database</td>
<td>Enter the name of the MongoDB database to be connected to.</td>
</tr>
<tr>
<td>Authentication mechanism</td>
<td>Among the mechanisms listed on the <strong>Authentication mechanism</strong> drop-down list, the <strong>NEGOTIATE</strong> one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using.</td>
</tr>
<tr>
<td>Set Authentication database</td>
<td>If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name</td>
</tr>
</tbody>
</table>
of this Authentication database in the **Authentication database** field that is displayed.

For further information about the MongoDB Authentication database, see [User Authentication database](#).

### Username and Password

DB user authentication data.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

Available when the **Use authentication** check box is selected.

If the security system you have selected from the **Authentication mechanism** drop-down list is Kerberos, you need to enter the **User principal**, the **Realm** and the **KDC server** fields instead of the **Username** and the **Password** fields.

### Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

### Execute command

Select this check box to enter MongoDB commands in the **Command** field for execution.

- **Command**: in this field, enter the command to be executed, if this command contains one single variable.

For example, if you need to construct the command

```json
{"isMaster": 1}
```

You need simply enter `isMaster` within quotation marks.

- **Construct command from keys and values**: if the command to be executed contains multiple
variables, select this check box and in the **Command keys and values** table, add the variables and their respective values to be used.

For example, if you need to construct the following command

```javascript
{ renameCollection : "<source_namespace>" ,
  to : "<target_namespace>" ,
  dropTarget : < true | false > }
```

You need to add three rows to the **Command keys and values** table and enter one variable-value pair to each row within quotation marks:

```
"renameCollection"
"old_name"
"to"
"new_name"
"dropTarget" "false"
```

- **Construct command from a JSON string**: if you want to directly enter the command to be used, select this check box and enter this command in the **JSON string command** field that is displayed. Only one command is allowed per **tCosmosDBRow**.

For example:

```
"{createIndexes: 'restaurants',
 indexes : [(key :
 {restaurant_id: 1}, name: 'id_index_2', unique: true})]"
```

Note that you must use single quotation marks to surround the string values used in the command and double quotation marks to surround the command itself.

For further information about the MongoDB commands you can use in this field, see https://docs.mongodb.org/manual/reference/command/.

### Advanced settings

| **Die on error** | This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows. |

### Usage

| **Usage rule** | **tCosmosDBRow** allows you to manipulate the Cosmos database through the MongoDB commands. |
tCouchbaseInput

Queries the documents from the Couchbase database.

**tCouchbaseInput Standard properties**

These properties are used to configure tCouchbaseInput running in the Standard Job framework.

The Standard tCouchbaseInput component belongs to the Databases family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bucket</strong></td>
<td>Enter, within double quotation marks, the name of the data bucket in the Couchbase database. If you are using Couchbase V5.0, this bucket name is the user name you have created in the Security tab of your Couchbase UI.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Provide the authentication credentials to a bucket. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. If you are using Couchbase V5.0, enter the same value you put in the <strong>Bucket</strong> field as password, because in Couchbase V5.0, no password is associated with a bucket. However, on Couchbase, you need to create a user with appropriate role to access the buckets. For further information about the access control and other important requirements on the Couchbase V5.0 side, see Couchbase V5.0 release note: Major Behavior Changes.</td>
</tr>
<tr>
<td><strong>Bootstrap nodes</strong></td>
<td>Enter the name or IP of the node to be bootstrapped by Couchbase SDK. As Couchbase recommends to specify multiple nodes to bootstrap, enter the names or IPs of these nodes in this field, separating them using commas (,). For further information about Couchbase bootstrapping, see How Couchbase SDKs connect to the cluster.</td>
</tr>
<tr>
<td><strong>Schema</strong> and <strong>Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. The schema of this component is read-only. The content column stores the documents to be used, the key column the IDs of these documents and the other columns the Couchbase technical information.</td>
</tr>
</tbody>
</table>
Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

Global Variables

| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |

Usage

| Usage rule | As a start component, **tCouchbaseInput** reads the documents from the Couchbase database. |
**tCouchbaseOutput**

Upserts documents in the Couchbase database based on the incoming flat data from preceding components.

This means that it adds a new document or replaces its value if it already exists.

**tCouchbaseOutput Standard properties**

These properties are used to configure tCouchbaseOutput running in the Standard Job framework.

The Standard tCouchbaseOutput component belongs to the Databases family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

### Basic settings

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Bucket**     | Enter, within double quotation marks, the name of the data bucket in the Couchbase database.  
If you are using Couchbase V5.0, this bucket name is the user name you have created in the Security tab of your Couchbase UI. |
| **Password**   | Provide the authentication credentials to a bucket.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.  
If you are using Couchbase V5.0, enter the same value you put in the Bucket field as password, because in Couchbase V5.0, no password is associated with a bucket. However, on Couchbase, you need to create a user with appropriate role to access the buckets.  
For further information about the access control and other important requirements on the Couchbase V5.0 side, see Couchbase V5.0 release note: Major Behavior Changes. |
| **Bootstrap nodes** | Enter the name or IP of the node to be bootstrapped by Couchbase SDK. As Couchbase recommends to specify multiple nodes to bootstrap, enter the names or IPs of these nodes in this field, separating them using commas (.).  
For further information about Couchbase bootstrapping, see How Couchbase SDKs connect to the cluster. |
| **Schema** and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. |
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.

Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

**Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

**Field to use as ID**

Enter, within double quotation marks, the name of the column from the schema of tCouchbaseOutput to provide IDs for the documents to be written to Couchbase.

**Die on error**

Select the check box to stop the execution of the Job when an error occurs.

Clear the check box to skip any rows on error and complete the process for error-free rows.

**Advanced settings**

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.
Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_SUCCESS: the number of rows successfully processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_REJECT: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Preceded by an input component, tCouchbaseOutput wraps flat data into documents for storage in the Couchbase database.</th>
</tr>
</thead>
</table>
tCouchDBClose

Disconnects a connection to a CouchDB server in order to release the occupied resources.

**tCouchDBClose Standard properties**

These properties are used to configure tCouchDBClose running in the Standard Job framework.

The Standard tCouchDBClose component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Component List</th>
<th>Select an active CouchDB connection to be closed.</th>
</tr>
</thead>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is generally used with other CouchDB components, particularly tCouchDBConnection.</th>
</tr>
</thead>
</table>

**Related Scenario**

For a scenario in which tCouchDBClose is used, see Scenario: Replicating data from the source database to the target database on page 534.
tCouchDBConnection

Opens a connection to a CouchDB server to be reused by the other CouchDB components. You can also configure replication parameters if a replication is triggered between the source database and the target database.

**tCouchDBConnection Standard properties**

These properties are used to configure tCouchDBConnection running in the Standard Job framework. The Standard tCouchDBConnection component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Version</td>
<td>Select the CouchDB version that you are using.</td>
</tr>
<tr>
<td>Server</td>
<td>Type in the IP address or hostname of the CouchDB server.</td>
</tr>
<tr>
<td>Port</td>
<td>Type in the listening port number of the CouchDB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Type in the database that you want to use.</td>
</tr>
<tr>
<td>Required authentication</td>
<td>Select this check box to provide credentials for the CouchDB authentication.</td>
</tr>
<tr>
<td></td>
<td>• Username: Type in the user name for the CouchDB authentication.</td>
</tr>
<tr>
<td></td>
<td>• Password: Type in the password for the CouchDB authentication.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Replicate target database</td>
<td>Select the check box to set the replication in the table that appears.</td>
</tr>
<tr>
<td></td>
<td><strong>Target DB name:</strong> Specify the target database to which the documents will be copied. It can be a local database name or a remote database URL.</td>
</tr>
<tr>
<td></td>
<td><strong>Continuous:</strong> Select this check box to continue a replication upon the server restart.</td>
</tr>
<tr>
<td></td>
<td><strong>Create target DB:</strong> Select this check box to create the target database if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>IS Canceled:</strong> Select this check box to cancel the existing replication between the specified source database and target database at the end of the Job.</td>
</tr>
</tbody>
</table>
Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component is generally used with other CouchDB components, particularly tCouchDBClose. |

Related scenario

For a scenario in which tCouchDBConnection is used, see Scenario: Replicating data from the source database to the target database on page 534.
**tCouchDBInput**

Extracts the desired JSON data out of a CouchDB database to make transformation to it, migrates it to another target format, or processes it before inserting it to the same database.

**tCouchDBInput Standard properties**

These properties are used to configure tCouchDBInput running in the Standard Job framework.

The Standard tCouchDBInput component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
• **View schema**: choose this option to view the schema only.  
• **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.  
The columns in the schema may vary depending on your configuration:  
• If you select the **Query by view** check box and the **Is reduce** check box at the same time and specify a group level after selecting the **Group** check box, only the **key** and **value** columns are available in the schema.  
• If you select the **Include docs** check box but do not select the **Is reduce** check box, the **id**, **key**, **value** and **JSONDoc** columns are available in the schema.  
• If you keep both the **Is reduce** check box and **Include docs** check box clear, the **id**, **key**, and **value** columns are available in the schema. |

<p>| Use existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. |</p>
<table>
<thead>
<tr>
<th>DB Version</th>
<th>Select the CouchDB version that you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Hostname or IP address of the CouchDB server.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of the CouchDB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Specify the database that you want to use.</td>
</tr>
<tr>
<td>Required authentication</td>
<td>Select this check box to provide credentials for the CouchDB authentication.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Username</strong>: Type in the user name for the CouchDB authentication.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Password</strong>: Type in the password for the CouchDB authentication.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td>Use trigger replication</td>
<td>Select this check box to trigger replication between databases.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning:</strong></td>
</tr>
<tr>
<td></td>
<td>This check box appears only if you do not select the <strong>Use an existing connection</strong> check box.</td>
</tr>
<tr>
<td>Target DB name</td>
<td>Specify the target database to which the documents will be copied. It can be a local database name or a remote database URL.</td>
</tr>
<tr>
<td>Continuous</td>
<td>Select this check box to continue a replication upon the server restart.</td>
</tr>
<tr>
<td>Create target DB</td>
<td>Select this check box to create the target database if it does not exist.</td>
</tr>
<tr>
<td>IS Canceled</td>
<td>Select this check box to cancel the existing replication between the specified source database and target database at the end of the Job.</td>
</tr>
<tr>
<td>Query by view</td>
<td>Select this check box to specify query conditions based on a view which involves one map function and one optional reduce function.</td>
</tr>
<tr>
<td>Design Document</td>
<td>Type in the name of the design document from which you want to read data.</td>
</tr>
<tr>
<td>Action on design document</td>
<td>Select the operation you want to perform on the design document of interest:</td>
</tr>
<tr>
<td></td>
<td>• <strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Drop and create design document</strong>: The design document is removed and created again.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Create design document</strong>: A new design document is created.</td>
</tr>
<tr>
<td>View</td>
<td>Type in the name of the view from which you want to read data.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Action on view</td>
<td>Select the operation you want to perform on the view of interest:</td>
</tr>
<tr>
<td>None</td>
<td>No operation is carried out.</td>
</tr>
<tr>
<td>Create view</td>
<td>A new view is created.</td>
</tr>
<tr>
<td>Map</td>
<td>Enter the map function in this field.</td>
</tr>
<tr>
<td>Reduce</td>
<td>Enter the reduce function in this field.</td>
</tr>
<tr>
<td>Start key</td>
<td>Type in the start key.</td>
</tr>
<tr>
<td>End key</td>
<td>Type in the end key.</td>
</tr>
<tr>
<td>Start key docid</td>
<td>Type in the start key document id.</td>
</tr>
<tr>
<td>End key docid</td>
<td>Type in the end key document id.</td>
</tr>
<tr>
<td>Is reduce</td>
<td>Select this check box to make the reduce function take effect.</td>
</tr>
<tr>
<td>Group</td>
<td>Select this check box to make the reduce function reduce to a set of distinct keys or to a single result row.</td>
</tr>
<tr>
<td>Group level</td>
<td>Enter the specific group level in this field after you select the Group check box.</td>
</tr>
<tr>
<td>Include docs</td>
<td>Select this check box to include the document which emitted each view entry.</td>
</tr>
<tr>
<td>Descending</td>
<td>Select this check box to reverse the query order.</td>
</tr>
</tbody>
</table>

- **Create design document if not exists**: A design document is created if it does not exist.
- **Drop design document if exists and create**: The design document is removed if it exists and gets created again.
### tCouchDBInput

<table>
<thead>
<tr>
<th>Add options</th>
<th>Select this check box to add more query options and define the parameters as needed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract JSON field</td>
<td>Select this check box to extract the desired JSON data based on XPath query.</td>
</tr>
<tr>
<td>JSON field</td>
<td>List of JSON fields to be extracted.</td>
</tr>
<tr>
<td><strong>Warning:</strong> Make sure you select a JSON field whose value is a JSON document. Otherwise, errors may occur unexpectedly.</td>
<td></td>
</tr>
<tr>
<td>Loop XPath query</td>
<td>Node within the JSON field, on which the loop is based.</td>
</tr>
</tbody>
</table>
| Mapping | **Schema output column:** schema defined to hold the data extracted from the JSON field.  
**XPath query:** XPath query to specify the node within the JSON field to be extracted.  
**Get Nodes:** select this check box if you need to get values from a nested node within the JSON field. |
| Limit | Enter the maximum number of rows to process. |
| Die on error | This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows. |

### Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

### Global Variables

| Global Variables | **NB_LINE:** the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

---

529
Usage

| Usage rule | This component always needs an output link. |

Related Scenario

For a scenario in which tCouchDBInput is used, see Scenario: Replicating data from the source database to the target database on page 534.
**tCouchDBOutput**

Loads JSON documents, writes data into or remove data from them and then saves the documents back to the database on a CouchDB server.

tCouchDBOutput receives data from the preceding component, and writes data into CouchDB.

**tCouchDBOutput Standard properties**

These properties are used to configure tCouchDBOutput running in the Standard Job framework.
The Standard tCouchDBOutput component belongs to the Big Data and the Databases families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

### Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click <a href="#">Edit schema</a> to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the <a href="#">Repository Content</a> window.</td>
<td></td>
</tr>
<tr>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see <a href="https://help.talend.com">Talend Studio User Guide</a>.</td>
<td></td>
</tr>
<tr>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <a href="https://help.talend.com">Talend Studio User Guide</a>. When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>].</td>
<td></td>
</tr>
<tr>
<td><strong>Sync columns</strong></td>
<td>Click this button to retrieve schema from the previous component connected in the Job.</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Use existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>DB Version</strong></td>
<td>Select the CouchDB version that you are using.</td>
</tr>
<tr>
<td><strong>Server</strong></td>
<td>Hostname or IP address of the CouchDB server.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of the CouchDB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database that you want to use.</td>
</tr>
</tbody>
</table>
| **Required authentication** | Select this check box to provide credentials for the CouchDB authentication.  
  • **Username**: Type in the user name for the CouchDB authentication.  
  • **Password**: Type in the password for the CouchDB authentication.  
  To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| **Use trigger replication** | Select this check box to trigger replication between databases.  
  **Warning:**  
  This check box appears only if you do not select the Use an existing connection check box. |
| **Target DB name** | Specify the target database to which the documents will be copied. It can be a local database name or a remote database URL. |
| **Continuous** | Select this check box to continue a replication upon the server restart. |
| **Create target DB** | Select this check box to create the target database if it does not exist. |
| **IS Canceled** | Select this check box to cancel the existing replication between the specified source database and target database at the end of the Job. |
| **Action on data** | On the data of interest, you can perform:  
  • **Insert**: Insert data into the database.  
  • **Update**: Update data in the database.  
  • **Upsert**: Insert the data if they do not exist or update the existing data.  
  • **Delete**: Remove the data from the database. |
<table>
<thead>
<tr>
<th><strong>Generate JSON Document</strong></th>
<th>Select this check box to generate a JSON document and configure the desired data structure for it.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key</strong></td>
<td>Select the key that you want to use from the list.</td>
</tr>
<tr>
<td><strong>Configure JSON Tree</strong></td>
<td>Click the [...] button to open the window for JSON tree configuration. For more information, see Configuring a JSON Tree on page 4040.</td>
</tr>
<tr>
<td><strong>Group by</strong></td>
<td>Customize the input columns based on which you want to group the data.</td>
</tr>
<tr>
<td><strong>Remove root node</strong></td>
<td>Select this check box to remove the root node.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| **Global Variables** | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. **NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer. **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

**Usage**

| **Usage rule** | This component is used as an output component and it always needs an incoming link. |
Scenario: Replicating data from the source database to the target database

This scenario applies only to a Talend solution with Big Data.

This scenario describes a Job that writes the information of books into a CouchDB database, replicates the data to the target database, and then displays some replicated information of interest on the console.

Linking the components

Procedure

1. Drop the following components from the Palette onto the design workspace: tCouchDBConnection, tFileInputDelimited, tCouchDBOutput, tCouchDBInput, tLogRow and tCouchDBClose.
2. Connect tFileInputDelimited to tCouchDBOutput using a Row > Main link.
3. Do the same to connect tCouchDBInput to tLogRow.
4. Connect tCouchDBConnection to tFileInputDelimited using a Trigger > OnSubjobOk link.
5. Do the same to connect tFileInputDelimited to tCouchDBInput and tCouchDBInput to tCouchDBClose.
6. Label the components to better identify their functions.
Configuring the components

Opening a CouchDB connection and triggering replication

Procedure

1. Double-click the **tCouchDBConnection** component to open its **Basic settings** view in the **Component** tab.

2. In the **Server** and **Port** fields, type in the connection details.

3. In the **Database** field, type in the name of the database you want to use: *bookstore_old* in this example.

4. Select the CouchDB version that you are using from the **DB Version** list.

5. Select the **Use trigger replication** check box.

6. In the **Replicate target database** area, click [*] to add one line for database replication settings.

7. Enter the name of the target database name: *bookstore_new* in this example.

8. Select the **Continuous** check box to continue the replication upon the server restart.

9. In this example, the target database does not exist. Select the **Create target DB** check box to create the target database.

10. Select the **Is Canceled** check box to cancel the replication between *bookstore_old* and *bookstore_new* at the end of the Job.

Reading the input data

Procedure

1. Double-click the **tFileInputDelimited** component to open its **Component** view.
2. Click the [...] button next to the **File name/Stream** field to browse to the file that you want to read data from. In this scenario, it is `D:/Input/bookstore.txt`. The file contains six columns: `_id`, `title`, `author`, `category`, `ISBN`, and `abstract._id;title;author;category;ISBN;abstract 00 1;Computer Networks: A Systems Approach;Larry L. Peterson, Bruce S. Davie;Computer Science;0123850606;This best-selling and classic book teaches you the key principles of computer networks with examples drawn from the real world of network and protocol design. 002;David Copperfield;Charles Dickens;Language&Literature;1555763227;This adaptation of the original story is presented in the format of a novel study, complete with exercises and vocabulary lists, and is geared to the language arts classes of grades 4 and 5. 003;Life of Pi;Yann Martel;Language&Literature;0547350651;The son of a zookeeper, Pi Patel has an encyclopedic knowledge of animal behavior and a fervent love of stories. 004;Les Miserables: Easyread Comfort Edition;Victor Hugo;Language&Literature;1425048250;Expressing the author's ideas about society, religion and politics, it is in the backdrop of Napoleonic Wars and ensuing years that the story unravels. Grace, moral philosophy, law and history of France are discussed. 005;Computer Security;Dieter Gollmann;Computer Science;0470741155;This text moves away from the 'multi-level' security approach to compare and evaluate design alternatives in computer security. 006;Advanced Database Systems;Carlo Zaniolo;Database;155860443X;This book, written by a team of leading specialists in their fields, introduces the research issues at the forefront of database technology and supports them with a variety of examples.

3. In the **Header** field, type in 1 so that the header of the file will be skipped.
4. Click **Edit schema** to define the data to pass on to the **tCouchDBOutput** component.

---

### Writing data into one database and replicating the data

**Procedure**

1. Double-click the **tCouchDBOutput** component to open its **Basic settings** view in the **Component** tab.

2. Click **Sync columns** to retrieve the schema from the preceding component.

3. Select the **Use an existing connection** check box. In this example, the replication is triggered when opening the CouchDB connection.

4. Select **Upsert** from the **Action on data** drop-down list.

---

### Extracting the replicated data of interest

**Procedure**

1. Double-click the **tCouchDBInput** component to open its **Component** view.
2. Click **Edit schema** to define the data structure to be read from the CouchDB database.

By default, the **Include docs** check box is selected, so the id, key, value and **jsonDoc** columns are available in the schema.

In this example, we define four columns to be extracted: id, title, author and category.
3. Enter the **Server** and **Port** information.

4. In the **Database** field, enter the name of the database from which the replicated data will be read. In this example, it is *bookstore_new*.

5. In the **Querying options** area, type in the start key and end key to set the range of the data to be read: ‘001’ and ‘006’ in this example.

6. Select the **Extract JSON field** check box to extract the desired data.

7. Select **jsonDoc** from the **JSON field** list.

8. In the **Mapping** area, click [+] to add items. Select the schema output column from the list and then type in the proper XPath query.

### Displaying the extracted data

**Procedure**

1. Double-click the **tLogRow** component to open its **Component** view.

2. Click **Edit schema** to define the data structure to be displayed on the console. In this example, we need to remove the **jsonDoc** column.

3. In the **Mode** area, select **Table (print values in cells of a table)**.

### Closing the CouchDB connection

**Procedure**

1. Double-click the **tCouchDBClose** component to open its **Component** view.
2. Select the connection to be closed from the **Component List**.

**Saving and executing the Job**

**Procedure**
1. Press **Ctrl+S** to save your Job.
2. Execute the Job by pressing **F6** or clicking **Run** on the **Run** tab.

```
The book information read from the replicated database is shown on the console.
```

The book information read from the replicated database is shown on the console.
**tCreateTable**

Creates a table for a specific type of database.

**tCreateTable Standard properties**

These properties are used to configure tCreateTable running in the Standard Job framework.

The Standard tCreateTable component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Select the type of the database. The connection properties may differ slightly according to the database type selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td></td>
<td>For an example about how to share a database connection across Job levels, see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB Version</th>
<th>Select the version of the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>The IP address or hostname of the database.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>The listening port number of the database.</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td><strong>Database name</strong></td>
<td>The name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>The name of the database schema.</td>
</tr>
<tr>
<td></td>
<td>This property is available for DB2, Exasol, Greenplum, Informix, MS SQL Server, Oracle, PostgresPlus, Postgresql, Redshift, Sybase, and Vertica database types.</td>
</tr>
<tr>
<td><strong>Access File</strong></td>
<td>The path to the Access database file.</td>
</tr>
<tr>
<td><strong>Firebird File</strong></td>
<td>The path to the Firebird database file.</td>
</tr>
<tr>
<td><strong>Interbase File</strong></td>
<td>The path to the Interbase database file.</td>
</tr>
<tr>
<td><strong>SQLite File</strong></td>
<td>The path to the SQLite database file.</td>
</tr>
<tr>
<td><strong>Running Mode</strong></td>
<td>Select the Server Mode that corresponds to your database setup.</td>
</tr>
<tr>
<td></td>
<td>This property is available only for the HSQLDb database type.</td>
</tr>
<tr>
<td><strong>Use TLS/SSL Sockets</strong></td>
<td>Select this check box to enable the security mode if required.</td>
</tr>
<tr>
<td></td>
<td>This property is available only for the HSQLDb database type.</td>
</tr>
<tr>
<td><strong>DB Alias</strong></td>
<td>The name of the database.</td>
</tr>
<tr>
<td></td>
<td>This property is available only for the HSQLDb database type.</td>
</tr>
<tr>
<td><strong>Framework Type</strong></td>
<td>Select the framework type for your database.</td>
</tr>
<tr>
<td></td>
<td>This property is available only for the JavaDb database type.</td>
</tr>
<tr>
<td><strong>DB Root Path</strong></td>
<td>Browse to your database root.</td>
</tr>
<tr>
<td></td>
<td>This property is available only for the JavaDb database type.</td>
</tr>
<tr>
<td><strong>ODBC Name</strong></td>
<td>The name of the ODBC database.</td>
</tr>
<tr>
<td><strong>Connection Type</strong></td>
<td>Select the Oracle database connection type.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Oracle SID</strong>: select this connection type to uniquely identify a particular database on a system.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Oracle Service</strong>: select this connection type to use the TNS alias that you give when you connect to the remote database.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Oracle OCI</strong>: select this connection type to use Oracle Call Interface with a set of C-language software APIs that provide an interface to the Oracle database.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Oracle Custom</strong>: select this connection type to access a clustered database.</td>
</tr>
</tbody>
</table>
### WALLET
Select this connection type to store credentials in an Oracle wallet.

### Username and Password
The database user authentication data.
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

### Table name
The name of the table to be created.

### Table Action
Select the action to be carried out on the table.
- **Create table**: the specified table doesn’t exist and gets created.
- **Create table if not exists**: the specified table is created if it does not exist.
- **Drop table if exits and create**: the table is removed if it already exists and gets created again.

### Case Sensitive
Select this check box to make the table/column name case sensitive.
This property is available only for the HSQLDb database type.

### Temporary Table
Select this check box if you want to save the created table temporarily.
This property is available only for the MySQL database type.

### Create
Select the type of the table to be created.
- **SET TABLE**: the table that does not allow duplicate rows.
- **MULTISET TABLE**: the table that allows duplicate rows.
This property is available only for the Teradata database type.

### Schema and Edit Schema
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.
Advanced settings

**Additional JDBC Parameters**

Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the Use an existing connection check box in the Basic settings.

This property is available for the AS/400 and MSSQL Server database types.

**Create projection**

Select this check box to create a projection.

This property is available only for the Vertica database type.

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Global Variables

**QUERY**

The query statement being processed. This is a Flow variable and it returns a string.

**ERROR_MESSAGE**

The error message generated by the component when an error occurs. This is an After variable and it returns a string.

Usage

**Usage rule**

This component can be used as a standalone component of a Job or Subjob.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.
Scenario: Creating new table in a Mysql Database

The Job described below aims at creating a table in a database, made of a dummy schema taken from a delimited file schema stored in the Repository. This Job is composed of a single component.

Procedure

Procedure

1. Drop a `tCreateTable` component from the Databases family in the Palette to the design workspace.
2. In the Basic settings view, and from the Database Type list, select Mysql for this scenario.

3. From the Table Action list, select Create table.
4. Select the Use Existing Connection check box only if you are using a dedicated DB connection component `tMysqlConnection` on page 2475. In this example, we won’t use this option.
5. In the Property type field, select Repository so that the connection fields that follow are automatically filled in. If you have not defined your DB connection metadata in the DB connection directory under the Metadata node, fill in the details manually as Built-in.
6. In the Table Name field, fill in a name for the table to be created.
7. If you want to retrieve the Schema from the Metadata (it doesn’t need to be a DB connection Schema metadata), select Repository then the relevant entry.
8. In any case (Built-in or Repository) click Edit Schema to check the data type mapping. Click Edit Schema to define the data structure.
9. Click the **Reset DB Types** button in case the DB type column is empty or shows discrepancies (marked in orange). This allows you to map any data type to the relevant DB data type. Then, click **OK** to validate your changes and close the dialog box.

10. Save your Job and press **F6** to execute it.

**Results**

The table is created empty but with all columns defined in the Schema.

```
SELECT * FROM `newcarstable`;
```

<table>
<thead>
<tr>
<th>ID_Owners</th>
<th>Reg_CAR</th>
<th>Make</th>
<th>Color</th>
<th>ID_Reseller</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**tCreateTemporaryFile**

Creates a temporary file in a specified directory. This component allows you to either keep the temporary file or delete it after the Job execution.

**tCreateTemporaryFile Standard properties**

These properties are used to configure tCreateTemporaryFile running in the Standard Job framework. The Standard tCreateTemporaryFile component belongs to the File family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove file when execution is over</td>
<td>Select this check box to delete the temporary file after the Job execution.</td>
</tr>
<tr>
<td>Use default temporary system directory</td>
<td>Select this check box to create the file in the default system temporary directory.</td>
</tr>
<tr>
<td>Directory</td>
<td>Specify the directory under which the temporary file will be created. This field is available only when the Use default temporary system directory check box is cleared.</td>
</tr>
<tr>
<td>Template</td>
<td>Enter the temporary file name which should contain the characters XXXX, such as talend_XXXX.</td>
</tr>
<tr>
<td>Suffix</td>
<td>Enter the filename extension of the temporary file.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
<tr>
<td>Statistics</td>
<td></td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILEPATH</td>
<td>the path where the file was created. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
<td></td>
</tr>
<tr>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
<td></td>
</tr>
</tbody>
</table>
Scenario: Creating a temporary file and writing data into it

This scenario describes a Job that creates a temporary file in the default system temporary directory, writes data into the file, and finally displays the data in the file on the console.
Adding and linking the components

Procedure

1. Create a new Job and add the following components by typing their names in the design workspace or dropping them from the Palette: a \texttt{tCreateTemporaryFile} component, a \texttt{tJava} component, a \texttt{tRowGenerator} component, a \texttt{tFileOutputDelimited} component, a \texttt{tFileInputDelimited} component, and a \texttt{tLogRow} component.

2. Connect \texttt{tRowGenerator} to \texttt{tFileOutputDelimited} using a \texttt{Row > Main} connection.

3. Do the same to connect \texttt{tFileInputDelimited} to \texttt{tLogRow}.

4. Connect \texttt{tCreateTemporaryFile} to \texttt{tJava} using a \texttt{Trigger > OnSubjobOk} connection.

5. Do the same to connect \texttt{tJava} to \texttt{tRowGenerator} and connect \texttt{tRowGenerator} to \texttt{tFileInputDelimited}.

Configuring the components

Creating the temporary file

Procedure

1. Double-click \texttt{tCreateTemporaryFile} to open its Basic settings view.
2. Select the **Remove file when execution is over** check box to delete the created temporary file after the Job execution.

3. Select the **Use default temporary system directory** check box to create the file in the default system temporary directory.

4. In the **Template** field, enter the temporary file name which should contain the characters `XXXX`. In this example, it is `talend_XXXX`.

5. In the **Suffix** field, enter the filename extension of the temporary file. In this example, it is `dat`.

6. Double-click **tJava** to open its **Basic settings** view.

7. In the **Code** field, enter the following code to display the default system temporary directory and the path to the temporary file that will be created on the console:

   ```java
   System.out.println("The default system temporary directory is:" +
   (String)System.getProperty("java.io.tmpdir"));
   System.out.println("The path to the temporary file is:" +
   (String)globalMap.get("tCreateTemporaryFile_1_FILEPATH"));
   ```

**Writing the data into the file**

**Procedure**

1. Double-click **tRowGenerator** to open its **RowGenerator Editor**.
2. Click the [+] button to add two columns: \textit{id} of Integer type and \textit{name} of String type. Then in the \textbf{Functions} column, select the predefined function \texttt{Numeric.sequence(String,int,int)} for \textit{id} and \texttt{TalendDataGenerator.getFirstName()} for \textit{name}.

3. In the \textbf{Number of Rows for RowGenerator} field, enter \texttt{5} to generate five rows.

4. Click \texttt{OK} to validate the changes and accept the propagation prompted by the pop-up dialog box.

5. Double-click \texttt{tFileOutputDelimited} to open its \textbf{Basic settings} view.

6. In the \textbf{File Name} field, press \texttt{Ctrl+Space} and from the global variable list displayed select \texttt{((String)globalMap.get("tCreateTemporaryFile_1_FILEPATH"))}.

\textbf{Reading the data from the file}

\textbf{Procedure}

1. Double-click \texttt{tFileInputDelimited} to open its \textbf{Basic settings} view.
2. In the **File name/Stream** field, press **Ctrl+Space** and from the global variable list displayed select `((String)globalMap.get("tCreateTemporaryFile_1_FILEPATH"));`.

3. Click the [...] button next to **Edit schema** and in the dialog box displayed define the schema by adding two columns: **id** of Integer type and **name** of String type.

4. Click **OK** to validate the changes and accept the propagation prompted by the pop-up dialog box.

5. Double-click **tLogRow** to open its **Basic settings** view.

6. In the **Mode** area, select **Table (print values in cells of a table)** to display the output data in a better way.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** or click **Run** on the **Run** tab to run the Job.

```
[statistics] connecting to socket on port 3406
[statistics] connected
The default system temporary directory is:
C:\Users\lena_li\AppData\Local\Temp\n
The path to the temporary file is:
C:\Users\lena_li\AppData\Local\Temp\talend_MHTI.dat

<table>
<thead>
<tr>
<th>tLogRow_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>--</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

[statistics] disconnected
```

The file *talend_MHTI.dat* is created under the default system temporary directory `C:\Users\lena_li\AppData\Local\Temp` during the Job execution, the five generated rows of data is written into it, then the file is deleted after the Job execution.
tDB2BulkExec

Executes the Insert action on the provided data and gains in performance during Insert operations to a DB2 database.

**tDB2BulkExec Standard properties**

These properties are used to configure tDB2BulkExec running in the Standard Job framework.

The Standard tDB2BulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*. 
<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Table Schema</strong></td>
<td>Name of the DB schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
</tbody>
</table>
| **Action on table** | On the table defined, you can perform one of the following operations:  
  - **None**: No operation is carried out.  
  - **Drop and create table**: The table is removed and created again.  
  - **Create table**: The table does not exist and gets created.  
  - **Create table if not exists**: The table is created if it does not exist.  
  - **Drop table if exists and create**: The table is removed if it already exists and created again.  
  - **Clear table**: The table content is deleted. |
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. |
| **Built-in** | You create the schema and store it locally for this component only. Related topic: see Talend Studio User Guide. |
| **Repository** | You have already created the schema and stored it in the Repository, hence can reuse it. Related topic: see Talend Studio User Guide. |
| **Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:** | |
| • **View schema**: choose this option to view the schema only. | |
| • **Change to built-in property**: choose this option to change the schema to Built-in for local changes. | |
| • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate |
the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

### Use Ingest Command


### Load From

Select the source of the data to be populated.
- **FILE**: loads data from a file.
- **PIPE**: loads data from a pipe.
- **FOLDER**: loads data from multiple files in a folder.

This list is available only when the Use Ingest Command check box is selected.

### Data File

Name of the file to be loaded.

**Warning:**

This file is located on the machine specified by the URI in the Host field so it should be on the same machine as the database server.

This field is not visible when PIPE or FOLDER is selected from the Load From drop-down list.

### Pipe Name

Enter the name of the pipe.

This field is available only when PIPE is selected from the Load From drop-down list.

### Folder

Specify the path to the folder holding the files to be loaded.

This field is available only when FOLDER is selected from the Load From drop-down list.

### Action on Data

On the data of the table defined, you can perform one of the following operations:
- **Insert**: Add new records to the table. If duplicates are found, Job stops.
- **Replace**: Add new records to the table. If an old record in the table has the same value as a new record for a PRIMARY KEY or a UNIQUE index, the old record is deleted before the new record is inserted.
- **Update**: Make changes to existing records.
- **Delete**: Remove the records that match the input data.
- **Merge**: Merge the input data to the table.

Delete and Merge are available only when the Use Ingest Command check box is selected.
<table>
<thead>
<tr>
<th>File Glob Pattern</th>
<th>Specify the global expression for the files to be loaded. This field is available only when FOLDER is selected from the Load From drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where Clause</td>
<td>Enter the WHERE clause to filter the data to be processed. This field is available only when update or delete is selected from the Action on Data drop-down list.</td>
</tr>
<tr>
<td>Custom Insert Values Clause</td>
<td>Select this check box and in the Insert Values Clause field displayed enter the VALUES clause for the insert operation. This check box is available only when the Use Ingest Command check box is selected and insert is selected from the Action on Data drop-down list.</td>
</tr>
<tr>
<td>Custom Update Set Clause</td>
<td>Select this check box and specify the SET clause for the update operation by completing the Set Mapping table. This check box is available only when the Use Ingest Command check box is selected and update is selected from the Action on Data drop-down list.</td>
</tr>
</tbody>
</table>
| Set Mapping                              | Complete this table to specify the SET clause for the update operation.  
  - **Column**: the name of the column. By default, the fields in the Column column are same as what they are in the schema.  
  - **Expression**: the expression for the corresponding column.  
  This table is available only when the Custom Update Set Clause check box is selected. |
| Merge Clause                             | Specify the MERGE clause for the merge operation. This table is available only when the Use Ingest Command check box is selected and merge is selected from the Action on Data drop-down list. |
| Content Format                           | Select the format of the input file, either Delimited or Positional. This list is available only when the Use Ingest Command check box is selected. |
| Delimited By                             | Enter the character that separates the fields in the delimited file. This field is available only when Delimited is selected from the Content Format drop-down list. |
| Optionally Enclosed By                   | Enter the character that encloses the string in the delimited file. This field is available only when Delimited is selected from the Content Format drop-down list. |
| Fixed Length | Enter the length (in bytes) of the record in the positional file.  
This field is available only when **Positional** is selected from the **Content Format** drop-down list. |
| --- | --- |
| Mapping | Complete this table to specify the mapping relationship between the source column and the DB2 table column.  
• **Column**: the name of the column. By default, the fields in the **Column** column are same as what they are in the schema.  
• **Is Table Column**: select the check box if the corresponding column is a table column.  
• **Start Position**: the starting position of the corresponding column.  
• **End Position**: the ending position of the corresponding column.  
The **Start Position** and **End Position** columns are available only when **Positional** is selected from the **Content Format** drop-down list.  
This table is available only when the **Use Ingest Command** check box is selected. |
| Script Generated Folder | Specify the directory under which the script file will be created.  
This field is available only when the **Use Ingest Command** check box is selected. |

**Advanced settings**

| Additional JDBC parameters | Specify additional connection properties for the DB connection you are creating.  
**Note:**  
You can set the encoding parameters through this field. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Field terminated by</td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td>Date Format</td>
<td>Use this field to define the way months and days are ordered.</td>
</tr>
<tr>
<td>Time Format</td>
<td>Use this field to define the way hours, minutes and seconds are ordered.</td>
</tr>
<tr>
<td>Timestamp Format</td>
<td>Use this field to define the way date and time are ordered.</td>
</tr>
<tr>
<td>Remove load pending</td>
<td>When the box is ticked, tables blocked in &quot;pending&quot; status following a bulk load are de-blocked.</td>
</tr>
</tbody>
</table>
| Load options | Click + to add data loading options:  
**Parameter**: select a loading parameter from the list. |
### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_LINE_UPDATED: the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_INSERTED: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_DELETED: the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>QUERY: the query statement processed. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This dedicated component offers performance and flexibility of DB2 query handling.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</td>
</tr>
<tr>
<td></td>
<td>The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.</td>
</tr>
</tbody>
</table>
Related scenarios

For tDB2BulkExec related topics, see:

- Scenario: Inserting transformed data in MySQL database on page 2537.
- Scenario: Truncating and inserting file data into an Oracle database on page 2752.
tDB2Close

Closes a transaction committed in the connected DB.

**tDB2Close Standard properties**

These properties are used to configure tDB2Close running in the Standard Job framework.

The Standard tDB2Close component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tDB2Connection</strong> component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

### Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with DB2 components, especially with <strong>tDB2Connection</strong> and <strong>tDB2Commit</strong>.</th>
</tr>
</thead>
</table>

| Dynamic settings | Click the [*] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario:
Related scenarios

No scenario is available for the Standard version of this component yet.
**tDB2Commit**

Commits in one go a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

tDB2Commit validates the data processed through the Job into the connected DB.

**tDB2Commit Standard properties**

These properties are used to configure tDB2Commit running in the Standard Job framework.

The Standard tDB2Commit component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tDB2Connection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Warning:**

If you want to use a Row > Main connection to link tDB2Commit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.

**Advanced settings**

| tStat Catcher Statistics                      | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule                                    | This component is more commonly used with other tDB2* components, especially with the tDB2Connection and tDB2Rollback components. |
## Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

---

### Related scenario

For **tDB2Commit** related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476
tDB2Connection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

**tDB2Connection Standard properties**

These properties are used to configure tDB2Connection running in the Standard Job framework.

The Standard tDB2Connection component belongs to the Databases and the ELT families.

The component in this framework is generally available.

⚠️ **Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Host name</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Table Schema</td>
<td>Name of the schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td>Encoding</td>
<td>Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td>Use or register a shared DB Connection</td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the <strong>Shared DB Connection Name</strong> field displayed, enter the name for the shared database</td>
</tr>
</tbody>
</table>
connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the Use dynamic job and Use an independent process to run subjob options of the tRunJob component. Using a shared connection together with a tRunJob component with either of these two options enabled will cause your Job to fail.

This check box is not available when the Specify a data source alias check box is selected.

### Specify a data source alias

Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

This check box is not visible when the Use or register a shared DB Connection check box is selected.

### Data source alias

Enter the alias of the data source created on the Talend Runtime side.

This field is available only when the Specify a data source alias check box is selected.

### Advanced settings

#### Additional JDBC parameters

Specify additional connection properties for the DB connection you are creating.

**Note:**

You can set the encoding parameters through this field.

#### Auto Commit

Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.
**tStatCatcher Statistics**

Select this check box to gather the job processing metadata at a Job level as well as at each component level.

**Usage**

**Usage rule**

This component is more commonly used with other tDB2* components, especially with the **tDB2Commit** and **tDB2Rollback** components.

**Related scenarios**

For **tDB2Connection** related scenario, see **tMysqlConnection** on page 2475
tDB2Input

Executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Main row link.

If double quotes exist in the column names of a table, the double quotation marks cannot be retrieved when retrieving the column. Therefore, it is recommended not to use double quotes in column names in a DB2 database table.

tDB2Input Standard properties

These properties are used to configure tDB2Input running in the Standard Job framework.
The Standard tDB2Input component belongs to the Databases family.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td></td>
<td>Note: When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
</tbody>
</table>
1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Schema</td>
<td>Name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>The schema is created and stored locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.
### Table name
Select the source table where to capture any changes made on data.

### Query type and Query
Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

### Specify a data source alias
Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

This check box is not available when the Use an existing connection check box is selected.

### Data source alias
Enter the alias of the data source created on the Talend Runtime side.

This field is available only when the Specify a data source alias check box is selected.

### Advanced settings

<table>
<thead>
<tr>
<th>Additional JDBC parameters</th>
<th>Specify additional connection properties for the DB connection you are creating.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>You can set the encoding parameters through this field.</td>
</tr>
<tr>
<td><strong>Trim all the String/Char columns</strong></td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</td>
</tr>
<tr>
<td><strong>Trim column</strong></td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

**NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

| Usage rule | This component covers all possible SQL queries for DB2 databases. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

For related topics, see:

See also the related topic in Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
tDB2Output

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.
tDB2Output writes, updates, makes changes or suppresses entries in a database.

tDB2Output Standard properties

These properties are used to configure tDB2Output running in the Standard Job framework.
The Standard tDB2Output component belongs to the Databases family.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related
to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components
on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
</tbody>
</table>
| Use an existing connection                    | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. Note: When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:
1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see Talend Studio User Guide. |
| Property type                                 | Either Built-in or Repository.                                             |
| Built-in                                      | No property data stored centrally.                                        |
**Repository:** Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Table schema</strong></td>
<td>Name of the DB schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td><strong>Default</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create a table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table if not exists</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop a table if exists and create</strong>: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear a table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td></td>
<td><strong>Truncate table</strong>: The table content is deleted. You do not have the possibility to rollback the operation.</td>
</tr>
<tr>
<td></td>
<td><strong>Truncate table with reuse storage</strong>: The table content is deleted. You do not have the possibility to rollback the operation. However, you can reuse the existing storage allocated to the table, even if the storage is considered empty.</td>
</tr>
</tbody>
</table>

**Warning:**

If you select the **Use an existing connection** check box, and then select **Truncate table** or **Truncate table with reuse storage** from the **Action on table** list, a commit statement will be invoked before the truncate operation.
because the truncate statement must be the first statement in a transaction.

<table>
<thead>
<tr>
<th>Action on data</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, Job stops.</td>
</tr>
<tr>
<td><strong>Update</strong>: Make changes to existing entries</td>
</tr>
<tr>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td><strong>Update or insert</strong>: Update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
</tr>
<tr>
<td><strong>Delete</strong>: Remove entries corresponding to the input flow.</td>
</tr>
</tbody>
</table>

**Warning:**

You must specify at least one column as a primary key on which the **Update** and **Delete** operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the update and delete operations. To do that: Select the **Use field options** check box and then in the **Key in update** column, select the check boxes next to the column name on which you want to base the update operation. Do the same in the **Key in delete** column for the deletion operation.

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Built-In:</th>
</tr>
</thead>
<tbody>
<tr>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Repository:</th>
</tr>
</thead>
<tbody>
<tr>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

<table>
<thead>
<tr>
<th>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Click Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
</tbody>
</table>
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Specify a data source alias
Select this check box and specify the alias of a data source created on the **Talend Runtime** side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in **Talend Runtime**. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

This check box is not available when the **Use an existing connection** check box is selected.

### Data source alias
Enter the alias of the data source created on the **Talend Runtime** side.

This field is available only when the **Specify a data source alias** check box is selected.

### Die on error
This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

### Advanced settings

#### Additional JDBC parameters
Specify additional connection properties for the DB connection you are creating.

**Note:**
You can set the encoding parameters through this field.

#### Commit every
Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.

#### Additional Columns
This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.

**Name:** Type in the name of the schema column to be altered or inserted as new column
**SQL expression**: Type in the SQL statement to be executed in order to alter or insert the relevant column data.

**Position**: Select **Before**, **Replace** or **After** following the action to be performed on the reference column.

**Reference column**: Type in a column of reference that the `tDBOutput` can use to place or replace the new or altered column.

**Use field options**
- Select this check box to customize a request, especially when there is double action on data.

**Convert columns and table names to uppercase**
- Select this check box to uppercase the names of the columns and the name of the table.

**Enable debug mode**
- Select this check box to display each step during processing entries in a database.

**Support null in “SQL WHERE” statement**
- Select this check box if you want to deal with the Null values contained in a DB table.
  
  **Note:**
  Make sure the Nullable check box is selected for the corresponding columns in the schema.

**Use Batch**
- Select this check box to activate the batch mode for data processing.
  
  **Note:**
  This check box is available only when you have selected the **Insert**, the **Update** or the **Delete** option in the **Action on data** field.

**Batch Size**
- Specify the number of records to be processed in each batch.
  
  This field appears only when the **Use batch mode** check box is selected.

**tStat Catcher Statistics**
- Select this check box to collect log data at the component level.

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE.UPDATED</td>
<td>the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE.INSERTED</td>
<td>the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE.DELETED</td>
<td>the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE.REJECTED</td>
<td>the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>
**ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

---

### Usage

#### Usage rule

This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a DB2 database. It also allows you to create a reject flow using a **Row > Rejects** link to filter data in error. For an example of **tMySqlOutput** in use, see *Scenario: Retrieving data in error with a Reject link* on page 2529.

#### Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489.

For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

---

### Related scenarios

For **tDB2Output** related topics, see

- *Scenario: Inserting a column and altering data using tMysqlOutput* on page 2521.
tDB2Rollback

Avoids to commit part of a transaction involuntarily.
tDB2Rollback cancels the transaction committed in the connected DB.

**tDB2Rollback Standard properties**

These properties are used to configure tDB2Rollback running in the Standard Job framework.
The Standard tDB2Rollback component belongs to the Databases family.
The component in this framework is generally available.

 petites notes:

This component is a specific version of a dynamic database connector. The properties related
to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components
on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tDB2Connection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other tDB2* components, especially with the tDB2Connection and tDB2Commit components.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</td>
</tr>
</tbody>
</table>
Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Related scenarios

For **tDB2Rollback** related scenario, see **Scenario: Rollback from inserting data in mother/daughter tables** on page 2480 of the **tMysqlRollback**.
tDB2Row

Acts on the actual DB structure or on the data (although without handling data) depending on the nature of the query and the database. The SQLBuilder tool helps you write easily your SQL statements.

tDB2Row is the specific component for this database query. It executes the SQL query stated onto the specified database. The row suffix means the component implements a flow in the job design although it doesn’t provide output.

**tDB2Row Standard properties**

These properties are used to configure tDB2Row running in the Standard Job framework.

The Standard tDB2Row component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td></td>
<td>For an example about how to share a database connection across Job levels, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

| Property type | Either Built-in or Repository |
| **Built-in** | No property data stored centrally. |
| **Repository** | Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved. |

| **Host** | Database server IP address |
| **Port** | Listening port number of DB server. |
| **Database** | Name of the database |
| **Username and Password** | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. |

**Built-in**: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.

Click **Edit schema** to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

<p>| <strong>Query type</strong> | Either Built-in or Repository. |
| <strong>Built-in</strong>: | Fill in manually the query statement or build it graphically using SQLBuilder |
| <strong>Repository</strong>: | Select the relevant query stored in the Repository. The Query field gets accordingly filled in. |</p>
<table>
<thead>
<tr>
<th><strong>Query</strong></th>
<th>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Specify a data source alias</strong></td>
<td>Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503. This check box is not available when the Use an existing connection check box is selected.</td>
</tr>
<tr>
<td><strong>Data source alias</strong></td>
<td>Enter the alias of the data source created on the Talend Runtime side. This field is available only when the Specify a data source alias check box is selected.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row &gt; Rejects link.</td>
</tr>
</tbody>
</table>

### Advanced settings

| **Additional JDBC parameters** | Specify additional connection properties for the DB connection you are creating.  

**Note:**  
You can set the encoding parameters through this field. |
| **Propagate QUERY’s recordset** | Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the use column list.  

**Note:**  
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of Object and this component is usually followed by tParseRecordSet. |
| **Commit every** | Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions. |
| **Use PreparedStatement** | Select this check box if you want to query the database using a PreparedStatement. In the Set Prepared Statement Parameter table, define the parameters represented by "?" in the SQL instruction of the Query field in the Basic Settings tab. |
Parameter Index: Enter the parameter position in the SQL instruction.

Parameter Type: Enter the parameter type.

Parameter Value: Enter the parameter value.

Note: This option is very useful if you need to execute the same query several times. Performance levels are increased.

tStatCatcher Statistics

Select this check box to collect log data at the component level.

Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>QUERY: the query statement being processed. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
</table>

Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-
Based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

For tDB2Row related topics, see:

- Scenario: Combining two flows for selective output on page 2558
- Procedure on page 625
- Scenario: Removing and regenerating a MySQL table index on page 2552.
tDB2SCD

Addresses Slowly Changing Dimension needs, reading regularly a source of data and logging the changes into a dedicated SCD table.

tDB2SCD reflects and tracks changes in a dedicated DB2 SCD table.

**tDB2SCD Standard properties**

These properties are used to configure tDB2SCD running in the Standard Job framework.

The Standard tDB2SCD component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in:</strong> No property data stored centrally.</td>
<td></td>
</tr>
</tbody>
</table>
**Repository**: Select the Repository file where properties are stored. The following fields are pre-filled in using fetched data.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Table Schema</strong></td>
<td>Name of the DB schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [... ] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
</tbody>
</table>
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  • **View schema**: choose this option to view the schema only.  
  • **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
  • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |

**Built-in**: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.

**SCD Editor**: The SCD editor helps to build and configure the data flow for slowly changing dimension outputs. For more information, see SCD management methodology on page 2566.

**Use memory saving Mode**: Select this check box to maximize system performance.
### Die on error
This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Additional JDBC parameters</strong></th>
<th>Specify additional connection properties for the DB connection you are creating.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>You can set the encoding parameters through this field.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>End date time details</strong></th>
<th>Specify the time value of the SCD end date time setting in the format of HH:mm:ss. The default value for this field is 12:00:00.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This field appears only when SCD Type 2 is used and Fixed year value is selected for creating the SCD end date.</td>
</tr>
</tbody>
</table>

| **Debug mode** | Select this check box to display each step during processing entries in a database. |

| **tStatCatcher Statistics** | Select this check box to collect log data at the component level. |

### Global Variables

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>NB_LINE_UPDATED</strong>: the number of rows updated. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_REJECTED</strong>: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as Output component. It requires an Input component and Row main link as input.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.  

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |
| Limitation | This component does not support using SCD type 0 together with other SCD types. |

## Related scenarios

For related topics, see tMysqlSCD on page 2563.
**tDB2SCDELT**

Addresses Slowly Changing Dimension needs through SQL queries (server-side processing mode), and logs the changes into a dedicated DB2 SCD table.

**tDB2SCDELT Standard properties**

These properties are used to configure tDB2SCDELT running in the Standard Job framework.

The Standard tDB2SCDELT component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally. Enter properties manually.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file where Properties are stored. The fields that come after are pre-filled in using the fetched data.</td>
<td></td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
<td></td>
</tr>
<tr>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
<td></td>
</tr>
</tbody>
</table>
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>The IP address of the database server.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>User authentication data for a dedicated database. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Source table</strong></td>
<td>Name of the input DB2 SCD table.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
</tbody>
</table>
| **Action on table** | Select to perform one of the following operations on the table defined:  
  - **None**: No action carried out on the table.  
  - **Drop and create table**: The table is removed and created again  
  - **Create table**: A new table gets created.  
  - **Create table if not exists**: A table gets created if it does not exist.  
  - **Clear table**: The table content is deleted. You have the possibility to rollback the operation.  
  - **Truncate table**: The table content is deleted. You don’t have the possibility to rollback the operation. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the Repository window. |
**Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Surrogate Key</th>
<th>Select the surrogate key column from the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation</td>
<td>Select the method to be used for the surrogate key generation. For more information regarding the creation methods, see <em>SCD management methodology</em> on page 2566.</td>
</tr>
<tr>
<td>Source Keys</td>
<td>Select one or more columns to be used as keys, to ensure the unicity of incoming data.</td>
</tr>
<tr>
<td>Source fields value include Null</td>
<td>Select this check box to allow the source columns to have Null values.</td>
</tr>
<tr>
<td></td>
<td>Note: The source columns here refer to the fields defined in the <em>SCD type 1 fields</em> and <em>SCD type 2 fields</em> tables.</td>
</tr>
<tr>
<td>Use SCD Type 1 fields</td>
<td>Use type 1 if tracking changes is not necessary. SCD Type 1 should be used for typos corrections for example. Select the columns of the schema that will be checked for changes.</td>
</tr>
<tr>
<td>Use SCD Type 2 fields</td>
<td>Use type 2 if changes need to be tracked down. SCD Type 2 should be used to trace updates for example. Select the columns of the schema that will be checked for changes.</td>
</tr>
<tr>
<td></td>
<td><strong>Start date</strong>: Adds a column to your SCD schema to hold the start date value. You can select one of the input schema columns as Start Date in the SCD table.</td>
</tr>
<tr>
<td></td>
<td><strong>End Date</strong>: Adds a column to your SCD schema to hold the end date value for the record. When the record is currently active, the End Date column shows a null value, or you can select <strong>Fixed Year value</strong> and fill it in with a fictive year to avoid having a null value in the End Date field.</td>
</tr>
<tr>
<td></td>
<td><strong>Log Active Status</strong>: Adds a column to your SCD schema to hold the true or false status value. This column helps to easily spot the active record.</td>
</tr>
<tr>
<td></td>
<td><strong>Log versions</strong>: Adds a column to your SCD schema to hold the version number of the record.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| Additional JDBC parameters | Specify additional connection properties for the DB connection you are creating. |
**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

**Usage**

| Usage rule | This component is used as an output component. It requires an input component and Row main link as input. |

| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable. For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |
Related Scenario

For related topics, see tDB2SCD on page 585 and tMysqlSCD on page 2563.
tDB2SP

Offers a convenient way to call the database stored procedures.

**tDB2SP Standard properties**

These properties are used to configure tDB2SP running in the Standard Job framework.

The Standard tDB2SP component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td></td>
<td>For an example about how to share a database connection across Job levels, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
</tbody>
</table>
| **Username and Password** | DB user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields.  
**Built-In:** You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.  
**Repository:** You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.  
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
- **View schema:** choose this option to view the schema only.  
- **Change to built-in property:** choose this option to change the schema to **Built-in** for local changes.  
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window. |
| **SP Name**   | Type in the exact name of the Stored Procedure                    |
| **Is Function / Return result in** | Check this box, if a value only is to be returned.  
Select on the list the schema column, the value to be returned is based on. |
| **Parameters** | Click the Plus button and select the various **Schema Columns** that will be required by the procedures. Note that the SP schema can hold more columns than there are parameters used in the procedure.  
Select the **Type** of parameter:  
**IN:** Input parameter  
**OUT:** Output parameter/return value |
**IN OUT**: Input parameters is to be returned as value, likely after modification through the procedure (function).

**RECORDSET**: Input parameters is to be returned as a set of values, rather than single value.

**Note:**
Check Scenario: Inserting data in mother/daughter tables on page 2476 if you want to analyze a set of records from a database table or DB query and return single records.

**Specify a data source alias**
Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

This check box is not available when the **Use an existing connection** check box is selected.

**Data source alias**
Enter the alias of the data source created on the Talend Runtime side.

This field is available only when the Specify a data source alias check box is selected.

**Advanced settings**

**Additional JDBC parameters**
Specify additional connection properties for the DB connection you are creating.

**Note:**
You can set the encoding parameters through this field.

**tStat Catcher Statistics**
Select this check box to collect log data at the component level.

**Usage**

**Usage rule**
This component is used as intermediary component. It can be used as start component but only input parameters are thus allowed.

**Dynamic settings**
Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.
The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Related scenarios

For related scenarios, see:

- Scenario: Retrieving personal information using a stored procedure on page 2452.
- Scenario: Using tMysqlSP to find a State Label using a stored procedure on page 2583.
- Scenario: Checking number format using a stored procedure on page 2809.
- Executing a stored procedure using tMDMSP on page 2190.

Check Scenario: Inserting data in mother/daughter tables on page 2476 as well if you want to analyze a set of records from a database table or DB query and return single records.
Dynamic database components

Talend provides a number of database components that allow you to change dynamically the type of database you want to work on. These components are available in the Database Common group under the Databases family of the Palette for standard data integration Jobs.

Each of these components has only one property, the Database list, on its Basic settings view for you to select the type of database of your interest.

For more information on these dynamic database components, see:

- tDBBulkExec on page 599
- tDBClose on page 600
- tDBCOLUMNList on page 602
- tDBCommit on page 603
- tDBConnection on page 605
- tDBINput on page 607
- tDBLastInsertId on page 609
- tDBOutput on page 610
- tDBOutputBulk on page 612
- tDBOutputBulkExec on page 613
- tDBRollback on page 614
- tDBRow on page 616
- tDBSCD on page 618
- tDBSCDELT on page 619
- tDBSP on page 620
- tDBTableList on page 621
tDBBulkExec

Offers gains in performance while executing the Insert operations on a database.

This component works with a variety of databases depending on your selection.

The tDBOutputBulk and tDBBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT statement used to feed a database the selected database type. These two steps are fused together in the tDBOutputBulkExec component, detailed in a separate section. The advantage of using two separate steps is that the data can be transformed before it is loaded in the database.

### tDBBulkExec Standard properties

These properties are used to configure tDBBulkExec running in the Standard Job framework.

The Standard tDBBulkExec component belongs to the Databases family.

The component in this framework is generally available.

#### Basic settings

This component serves as an entry point for the following databases. To configure this component, select a type of database from the Database list and click Apply on its Basic settings view. For more information about specific database properties, see the relevant documentation:

- Access (tAccessBulkExec on page 63)
- Amazon (tRedshiftBulkExec on page 3113)
- Greenplum (tGreenplumBulkExec on page 1318)
- IBM DB2 (tDB2BulkExec on page 554)
- Informix (tInformixBulkExec on page 1714)
- Ingres (tIngresBulkExec on page 1758)
- Microsoft SQL Server (tMSSqlBulkExec on page 2394)
- MySQL (tMysqlBulkExec on page 2461)
- Netezza (tNetezzaBulkExec on page 2674)
- Oracle (tOracleBulkExec on page 2747)
- ParAccel (tParAccelBulkExec on page 2879)
- PostgreSQL (tPostgresqlBulkExec on page 3063)
- PostgresPlus (tPostgresPlusBulkExec on page 3023)
- Sybase (ASE and IQ) (tSybaseBulkExec on page 3793)
- Sybase IQ (tSybaseIQBulkExec on page 3808)
- Vertica (tVerticaBulkExec on page 3961)
tDBClose

Closes the transaction committed in a connected database.
This component works with a variety of databases depending on your selection.

**tDBClose Standard properties**

These properties are used to configure tDBClose running in the Standard Job framework.
The Standard tDBClose component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

This component serves as an entry point for the following databases. To configure this component, select a type of database from the Database list and click Apply on its Basic settings view. For more information about specific database properties, see the relevant documentation:

- Access ([Access](#))
- Amazon Aurora ([Amazon Aurora](#))
- Amazon Mysql ([Amazon Mysql](#))
- Amazon Oracle ([Amazon Oracle](#))
- Amazon Redshift ([Amazon Redshift](#))
- AS400 ([AS400](#))
- Firebird ([Firebird](#))
- Greenplum ([Greenplum](#))
- IBM DB2 ([IBM DB2](#))
- Exasol ([Exasol](#))
- Informix ([Informix](#))
- Ingres ([Ingres](#))
- Interbase ([Interbase](#))
- JDBC ([JDBC](#))
- MemSQL ([MemSQL](#))
- Microsoft SQL Server ([Microsoft SQL Server](#))
- MySQL ([MySQL](#))
- Netezza ([Netezza](#))
- Oracle ([Oracle](#))
- ParAccel ([ParAccel](#))
- PostgreSQL ([PostgreSQL](#))
• PostgresPlus (tPostgresPlusClose on page 3027)
• SAP Hana (tSAPHanaClose on page 3456)
• SQLite (tSQLiteClose on page 3634)
• Snowflake (tSnowflakeClose on page 3542)
• Sybase (ASE and IQ) (tSybaseClose on page 3798)
• Teradata (tTeradataClose on page 3862)
• Vertica (tVerticaClose on page 3967)
tDBColumnList

Iterates on all columns of a given database table and lists column names.
This component works with a variety of databases depending on your selection.

**tDBColumnList Standard properties**

These properties are used to configure tDBColumnList running in the Standard Job framework.
The Standard tDBColumnList component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

This component serves as an entry point for the following databases. To configure this component,
select a type of database from the **Database** list and click **Apply** on its **Basic settings** view. For more
information about specific database properties, see the relevant documentation:

- Microsoft SQL Server (**tMSSqlColumnList** on page 2401)
- MySQL (**tMysqlColumnList** on page 2468)
tDBCommit

Validates the data processed through the Job into the connected database.
This component works with a variety of databases depending on your selection.

tDBCommit Standard properties

These properties are used to configure tDBCommit running in the Standard Job framework.
The Standard tDBCommit component belongs to the Databases family.
The component in this framework is generally available.

Basic settings

This component serves as an entry point for the following databases. To configure this component, select a type of database from the Database list and click Apply on its Basic settings view. For more information about specific database properties, see the relevant documentation:

- Access (tAccessCommit on page 69)
- Amazon Aurora (tAmazonAuroraCommit on page 134)
- Amazon Mysql (tAmazonMysqlCommit on page 174)
- Amazon Oracle (tAmazonOracleCommit on page 197)
- AS400 (tAS400Commit on page 227)
- Amazon Redshift (tRedshiftCommit on page 3131)
- FireBird (tFirebirdCommit on page 1182)
- Greenplum (tGreenplumCommit on page 1324)
- IBM DB2 (tDB2Commit on page 563)
- Exasol (tEXACommit on page 899)
- Informix (tInformixCommit on page 1721)
- Ingres (tIngresCommit on page 1764)
- Interbase (tInterbaseCommit on page 1798)
- JDBC (tJDBCCCommit on page 1867)
- Microsoft SQL Server (tMSSqlCommit on page 2404)
- MySQL (tMysqlCommit on page 2473)
- Netezza (tNetezzaCommit on page 2680)
- Oracle (tOracleCommit on page 2758)
- ParAccel (tParAccelCommit on page 2886)
- PostgreSQL (tPostgresqlCommit on page 3069)
- PostgresPlus (tPostgresPlusCommit on page 3029)
• SAPHana (tSAPHanaCommit on page 3457)
• SQLite (tSQLiteCommit on page 3636)
• Sybase (ASE and IQ) (tSybaseCommit on page 3800)
• Teradata (tTeradataCommit on page 3864)
• VectorWise (tVectorWiseCommit on page 3941)
• Vertica (tVerticaCommit on page 3969)
tDBConnection

Opens a connection to a database to be reused in the subsequent subjob or subjobs.
This component works with a variety of databases depending on your selection.

**tDBConnection Standard properties**

These properties are used to configure tDBConnection running in the Standard Job framework.
The Standard tDBConnection component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

This component serves as an entry point for the following databases. To configure this component,
select a type of database from the **Database** list and click **Apply** on its **Basic settings** view. For more
information about specific database properties, see the relevant documentation:

- Access ([tAccessConnection](#))
- Amazon Aurora ([tAmazonAuroraConnection](#))
- Amazon Mysql ([tAmazonMysqlConnection](#))
- Amazon Oracle ([tAmazonOracleConnection](#))
- Amazon Redshift ([tRedshiftConnection](#))
- AS400 ([tAS400Connection](#))
- Exasol ([tEXAConnection](#))
- FireBird ([tFirebirdConnection](#))
- Greenplum ([tGreenplumConnection](#))
- IBM DB2 ([tDB2Connection](#))
- Informix ([tInformixConnection](#))
- Ingres ([tIngresConnection](#))
- Interbase ([tInterbaseConnection](#))
- JDBC ([tJDBCConnection](#))
- MemSQL ([tMemSQLConnection](#))
- Microsoft SQL Server ([tMSSqlConnection](#))
- MySQL ([tMysqlConnection](#))
- Netezza ([tNetezzaConnection](#))
- Oracle ([tOracleConnection](#))
- ParAccel ([tParAccelConnection](#))
- PostgreSQL ([tPostgresqlConnection](#))
• PostgresPlus (tPostgresPlusConnection on page 3031)
• SAPHana (tSAPHanaConnection on page 3459)
• SQLite (tSQLiteConnection on page 3638)
• Snowflake (tSnowflakeConnection on page 3543)
• Sybase (ASE and IQ) (tSybaseConnection on page 3802)
• Teradata (tTeradataConnection on page 3866)
• VectorWise (tVectorWiseConnection on page 3943)
• Vertica (tVerticaConnection on page 3971)
tDBInput

Extracts data from a database.
This component works with a variety of databases depending on your selection.

**tDBInput Standard properties**

These properties are used to configure tDBInput running in the Standard Job framework.
The Standard tDBInput component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

This component serves as an entry point for the following databases. To configure this component, select a type of database from the Database list and click Apply on its Basic settings view. For more information about specific database properties, see the relevant documentation:

- Access (tAccessInput on page 77)
- Amazon Aurora (tAmazonAuroraInput on page 139)
- Amazon Mysql (tAmazonMysqlInput on page 179)
- Amazon Oracle (tAmazonOracleInput on page 202)
- Amazon Redshift (tRedshiftInput on page 3136)
- AS400 (tAS400Input on page 231)
- Exasol (tEXAInput on page 904)
- FireBird (tFirebirdInput on page 1186)
- Greenplum (tGreenplumInput on page 1334)
- IBM DB2 (tDB2Input on page 568)
- Informix (tInformixInput on page 1726)
- Ingres (tIngresInput on page 1768)
- Interbase (tInterbaseInput on page 1802)
- JDBC (tJDBCInput on page 1872)
- MemSQL (tMemSQLInput on page 2227)
- Microsoft SQL Server (tMSSqlInput on page 2414)
- MySQL (tMysqlInput on page 2488)
- Netezza (tNetezzaInput on page 2684)
- Oracle (tOracleInput on page 2764)
- ParAccel (tParAccelInput on page 2890)
- PostgreSQL (tPostgresqlInput on page 3074)
• PostgresPlus (tPostgresPlusInput on page 3033)
• SAP Hana (tSAPHanaInput on page 3461)
• SAS (tSasInput on page 3476)
• SQLite (tSQLiteInput on page 3640)
• Snowflake (tSnowflakeInput on page 3545)
• Sybase (ASE and IQ) (tSybaseInput on page 3804)
• Teradata (tTeradataInput on page 3878)
• VectorWise (tVectorWiseInput on page 3945)
• Vertica (tVerticalInput on page 3974)
tDBLastInsertId

Obtains the primary key value of the record that was last inserted in a database table by a user.
This component works with a variety of databases depending on your selection.

**tDBLastInsertId Standard properties**

These properties are used to configure tDBLastInsertId running in the Standard Job framework.
The Standard tDBLastInsertId component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

This component serves as an entry point for the following databases. To configure this component, select a type of database from the **Database** list and click **Apply** on its **Basic settings** view. For more information about specific database properties, see the relevant documentation:

- AS400 (**tAS400LastInsertId** on page 239)
- Microsoft SQL Server (**tMSSqlLastInsertId** on page 2419)
- MySQL (**tMysqlLastInsertId** on page 2508)
tDBOutput

Writes, updates, makes changes or suppresses entries in a database.
This component works with a variety of databases depending on your selection.

**tDBOutput Standard properties**

These properties are used to configure tDBOutput running in the Standard Job framework. The Standard tDBOutput component belongs to the Databases family. The component in this framework is generally available.

**Basic settings**

This component serves as an entry point for the following databases. To configure this component, select a type of database from the Database list and click Apply on its Basic settings view. For more information about specific database properties, see the relevant documentation:

- Access (tAccessOutput on page 81)
- Amazon Aurora (tAmazonAuroraOutput on page 149)
- Amazon Mysql (tAmazonMysqlOutput on page 183)
- Amazon Oracle (tAmazonOracleOutput on page 206)
- Amazon Redshift (tRedshiftOutput on page 3145)
- AS400 (tAS400Output on page 241)
- Exasol (tEXAOutput on page 908)
- FireBird (tFirebirdOutput on page 1190)
- Greenplum (tGreenplumOutput on page 1338)
- IBM DB2 (tDB2Output on page 572)
- Informix (tInformixOutput on page 1729)
- Ingres (tIngresOutput on page 1771)
- Interbase (tInterBaseOutput on page 1806)
- JDBC (tJDBCOutput on page 1876)
- MemSQL (tMemSQLOutput on page 2238)
- Microsoft SQL Server (tMSSqlOutput on page 2422)
- MySQL (tMysqlOutput on page 2515)
- Netezza (tNetezzaOutput on page 2695)
- Oracle (tOracleOutput on page 2772)
- ParAccel (tParAccelOutput on page 2894)
- PostgreSQL (tPostgresqlOutput on page 3078)
- PostgresPlus (tPostgresPlusOutput on page 3037)
- SAPHana (tSAPHanaOutput on page 3465)
- SAS (tSasOutput on page 3479)
- SQLite (tSQLiteOutput on page 3645)
- Snowflake (tSnowflakeOutput on page 3553)
- Sybase (ASE and IQ) (tSybaseOutput on page 3824)
- Teradata (tTeradataOutput on page 3885)
- VectorWise (tVectorWiseOutput on page 3949)
- Vertica (tVerticaOutput on page 3978)
tDBOutputBulk

Writes a file with columns based on the defined delimiter and the standards of the selected database type.

This component works with a variety of databases depending on your selection.

The tDBOutputBulk and tDBBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT statement used to feed a database the selected database type. These two steps are fused together in the tDBOutputBulkExec component, detailed in a separate section. The advantage of using two separate steps is that the data can be transformed before it is loaded in the database.

tDBOutputBulk Standard properties

These properties are used to configure tDBOutputBulk running in the Standard Job framework.

The Standard tDBOutputBulk component belongs to the Databases family.

The component in this framework is generally available.

Basic settings

This component serves as an entry point for the following databases. To configure this component, select a type of database from the Database list and click Apply on its Basic settings view. For more information about specific database properties, see the relevant documentation:

- Access (tAccessOutputBulk on page 87)
- Amazon Redshift (tRedshiftOutputBulk on page 3151)
- Greenplum (tGreenplumOutputBulk on page 1344)
- Informix (tInformixOutputBulk on page 1735)
- Ingres (tIngresOutputBulk on page 1776)
- Microsoft SQL Server (tMSSqlOutputBulk on page 2430)
- MySQL (tMysqlOutputBulk on page 2535)
- Oracle (tOracleOutputBulk on page 2779)
- ParAccel (tParAccelOutputBulk on page 2900)
- PostgreSQL (tPostgresqlOutputBulk on page 3085)
- PostgresPlus (tPostgresPlusOutputBulk on page 3043)
- Sybase (ASE and IQ) (tSybaseOutputBulk on page 3830)
- Vertica (tVerticaOutputBulk on page 3985)
tDBOutputBulkExec

Executes the Insert action in a database.

This component works with a variety of databases depending on your selection.

The tDBOutputBulk and tDBBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT statement used to feed a database the selected database type. These two steps are fused together in the tDBOutputBulkExec component, detailed in a separate section. The advantage of using two separate steps is that the data can be transformed before it is loaded in the database.

tDBOutputBulkExec Standard properties

These properties are used to configure tDBOutputBulkExec running in the Standard Job framework.

The Standard tDBOutputBulkExec component belongs to the Databases family.

The component in this framework is generally available.

Basic settings

This component serves as an entry point for the following databases. To configure this component, select a type of database from the Database list and click Apply on its Basic settings view. For more information about specific database properties, see the relevant documentation:

- Access (tAccessOutputBulk on page 87)
- Amazon Redshift (tRedshiftOutputBulk on page 3151)
- Greenplum (tGreenplumOutputBulk on page 1344)
- Informix (tInformixOutputBulk on page 1735)
- Ingres (tIngresOutputBulk on page 1776)
- Microsoft SQL Server (tMSSqlOutputBulk on page 2430)
- MySQL (tMysqlOutputBulk on page 2535)
- Oracle (tOracleOutputBulk on page 2779)
- ParAccel (tParAccelOutputBulk on page 2900)
- PostgreSQL (tPostgresqlOutputBulk on page 3085)
- PostgresPlus (tPostgresPlusOutputBulk on page 3043)
- Sybase (ASE and IQ) (tSybaseOutputBulk on page 3830)
- Vertica (tVerticaOutputBulk on page 3985)
tDBRollback

Cancels the transaction commit in a connected database to avoid committing part of a transaction involuntarily.

This component works with a variety of databases depending on your selection.

**tDBRollback Standard properties**

These properties are used to configure tDBRollback running in the Standard Job framework.

The Standard tDBRollback component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

This component serves as an entry point for the following databases. To configure this component, select a type of database from the Database list and click Apply on its Basic settings view. For more information about specific database properties, see the relevant documentation:

- Access ([tAccessRollback](#) on page 94)
- Amazon Aurora ([tAmazonAuroraRollback](#) on page 156)
- Amazon Mysql ([tAmazonMysqlRollback](#) on page 189)
- Amazon Oracle ([tAmazonOracleRollback](#) on page 212)
- Amazon Redshift ([tRedshiftRollback](#) on page 3161)
- AS400 ([tAS400Rollback](#) on page 246)
- Exasol ([tEXARollback](#) on page 914)
- FireBird ([tFirebirdRollback](#) on page 1196)
- Greenplum ([tGreenplumRollback](#) on page 1351)
- IBM DB2 ([tDB2Rollback](#) on page 578)
- Informix ([tInformixRollback](#) on page 1744)
- Ingres ([tIngresRollback](#) on page 1786)
- Interbase ([tInterbaseRollback](#) on page 1812)
- JDBC ([tJDBCRollback](#) on page 1882)
- Microsoft SQL Server ([tMSSqlRollback](#) on page 2438)
- MySQL ([tMysqlRollback](#) on page 2546)
- Netezza ([tNetezzaRollback](#) on page 2701)
- Oracle ([tOracleRollback](#) on page 2788)
- ParAccel ([tParAccelRollback](#) on page 2907)
- PostgreSQL ([tPostgresqlRollback](#) on page 3092)
• PostgresPlus (tPostgresPlusRollback on page 3049)
• SAPHana (tSAPHanaRollback on page 3471)
• SQLite (tSQLiteRollback on page 3650)
• Sybase (ASE and IQ) (tSybaseRollback on page 3838)
• Teradata (tTeradataRollback on page 3892)
• VectorWise (tVectorWiseRollback on page 3955)
• Vertica (tVerticaRollback on page 3994)
tDBRow

Executes the stated SQL query onto a database.
This component works with a variety of databases depending on your selection.

**tDBRow Standard properties**

These properties are used to configure tDBRow running in the Standard Job framework. The Standard tDBRow component belongs to the Databases family. The component in this framework is generally available.

**Basic settings**

This component serves as an entry point for the following databases. To configure this component, select a type of database from the Database list and click Apply on its Basic settings view. For more information about specific database properties, see the relevant documentation:

- Access (tAccessRow on page 96)
- Amazon Mysql (tAmazonMysqlRow on page 191)
- Amazon Oracle (tAmazonOracleRow on page 214)
- Amazon Redshift (tRedshiftRow on page 3163)
- AS400 (tAS400Row on page 248)
- Exasol (tEXARow on page 916)
- FireBird (tFirebirdRow on page 1198)
- Greenplum (tGreenplumRow on page 1353)
- IBM DB2 (tDB2Row on page 580)
- Informix (tInformixRow on page 1746)
- Ingres (tIngresRow on page 1788)
- Interbase (tInterbaseRow on page 1814)
- JDBC (tJDBCRow on page 1884)
- MemSQL (tMemSQLRow on page 2243)
- Microsoft SQL Server (tMSSqlRow on page 2440)
- MySQL (tMysqlRow on page 2548)
- Netezza (tNetezzaRow on page 2703)
- Oracle (tOracleRow on page 2790)
- ParAccel (tParAccelRow on page 2909)
- PostgreSQL (tPostgresqlRow on page 3094)
- PostgresPlus (tPostgresPlusRow on page 3051)
• SAPHana (tSAPHanaRow on page 3472)
• SQLite (tSQLiteRow on page 3652)
• Snowflake (tSnowflakeRow on page 3557)
• Sybase (ASE and IQ) (tSybaseRow on page 3840)
• Teradata (tTeradataRow on page 3894)
• VectorWise (tVectorWiseRow on page 3957)
• Vertica (tVerticaRow on page 3996)
tDBSCD

Reflects and tracks changes in a dedicated database SCD table.
This component works with a variety of databases depending on your selection.

**tDBSCD Standard properties**

These properties are used to configure tDBSCD running in the Standard Job framework.
The Standard tDBSCD component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

This component serves as an entry point for the following databases. To configure this component, select a type of database from the **Database** list and click **Apply** on its **Basic settings** view. For more information about specific database properties, see the relevant documentation:

- Greenplum (tGreenplumSCD on page 1358)
- IBM DB2 (tDB2SCD on page 585)
- Informix (tInformixSCD on page 1750)
- Ingres (tIngresSCD on page 1792)
- Microsoft SQL Server (tMSSqlSCD on page 2445)
- MySQL (tMysqlSCD on page 2563)
- Netezza (tNetezzaSCD on page 2708)
- Oracle (tOracleSCD on page 2795)
- ParAccel (tParAccelSCD on page 2914)
- PostgreSQL (tPostgresqlSCD on page 3099)
- PostgresPlus (tPostgresPlusSCD on page 3055)
- Sybase (ASE and IQ) (tSybaseSCD on page 3845)
- Teradata (tTeradataSCD on page 3899)
- Vertica (tVerticaSCD on page 4001)
tDBSCDELT

Reflects and tracks changes in a dedicated SCD table through SQL queries. 
This component works with a variety of databases depending on your selection.

**tDBSCDELT Standard properties**

These properties are used to configure tDBSCDELT running in the Standard Job framework. 
The Standard tDBSCDELT component belongs to the Databases family. 
The component in this framework is generally available.

**Basic settings**

This component serves as an entry point for the following databases. To configure this component, select a type of database from the **Database** list and click **Apply** on its **Basic settings** view. For more information about specific database properties, see the relevant documentation:

- IBM DB2 (tDB2SCDELT on page 589)
- MySQL (tMysqlSCDELT on page 2577)
- Oracle (tOracleSCDELT on page 2800)
- PostgreSQL (tPostgresqlSCDELT on page 3103)
- PostgresPlus (tPostgresPlusSCDELT on page 3059)
- Sybase (ASE and IQ) (tSybaseSCDELT on page 3849)
- Teradata (tTeradataSCDELT on page 3903)
**tDBSP**

Calls a database stored procedure.

This component works with a variety of databases depending on your selection.

**tDBSP Standard properties**

These properties are used to configure tDBSP running in the Standard Job framework.

The Standard tDBSP component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

This component serves as an entry point for the following databases. To configure this component, select a type of database from the Database list and click Apply on its Basic settings view. For more information about specific database properties, see the relevant documentation:

- IBM DB2 ([tDB2SP on page 594](#))
- Informix ([tInformixSP on page 1754](#))
- JDBC ([tJDBCSP on page 1901](#))
- Microsoft SQL Server ([tMSSqlSP on page 2449](#))
- MySQL ([tMysqlSP on page 2581](#))
- Oracle ([tOracleSP on page 2805](#))
- Sybase (ASE and IQ) ([tSybaseSP on page 3854](#))
**tDBTableList**

Lists the names of specified database tables using a SELECT statement based on a WHERE clause. This component works with a variety of databases depending on your selection.

**tDBTableList Standard properties**

These properties are used to configure tDBSP running in the Standard Job framework. The Standard tDBSP component belongs to the Databases family. The component in this framework is generally available.

**Basic settings**

This component serves as an entry point for the following databases. To configure this component, select a type of database from the Database list and click Apply on its Basic settings view. For more information about specific database properties, see the relevant documentation:

- Microsoft SQL Server (tMSSqlTableList on page 2459)
- MySQL (tMysqlTableList on page 2587)
- Oracle (tOracleTableList on page 2813)
tDBSQLRow

Acts on the actual DB structure or on the data (although without handling data) depending on
the nature of the query and the database. The SQLBuilder tool helps you write easily your SQL
statements.

tDBSQLRow is the generic component for database query. It executes the SQL query stated onto t
he specified database. The row suffix means the component implements a flow in the job design
although it does not provide output. For performance reasons, specific DB component should always
be preferred to the generic component.

To use this component, relevant DBMSs' ODBC drivers should be installed and the corresponding
ODBC connections should be configured via the database connection configuration wizard.

tDBSQLRow Standard properties

These properties are used to configure tDBSQLRow running in the Standard Job framework.
The Standard tDBSQLRow component belongs to the Databases family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Datasource</td>
<td>Name of the data source defined via the database connection configuration wizard.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td></td>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Name of the source table where changes made to data should be captured.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: Fill in manually the query statement or build it graphically using SQLBuilder</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</td>
<td></td>
</tr>
<tr>
<td>Query</td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Additional JDBC parameters</th>
<th>Specify additional connection properties for the database connection you are creating.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>You can set the encoding parameters through this field.</td>
</tr>
</tbody>
</table>

| Propagate QUERY's recordset | Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the use column list. |

<table>
<thead>
<tr>
<th>Use PreparedStatement</th>
<th>Select this check box if you want to query the database using a PreparedStatement. In the <strong>Set PreparedStatement Parameter</strong> table, define the parameters represented by &quot;?&quot; in the SQL instruction of the <strong>Query</strong> field in the Basic Settings tab.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter Index</td>
<td>Enter the parameter position in the SQL instruction.</td>
</tr>
</tbody>
</table>
Parameter Type: Enter the parameter type.
Parameter Value: Enter the parameter value.

Note:
This option is very useful if you need to execute the same query several times. Performance levels are increased.

Commit every
Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.

tStatCatcher Statistics
Select this check box to collect log data at the component level.

Global Variables

Global Variables

QUERY: the query statement being processed. This is a Flow variable and it returns a string.

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule
This component offers the flexibility of the DB query and covers all possible SQL queries.

Note that the relevant DBRow component should be preferred according to your DBMSs. Most of the DBMSs have their specific DBRow components.

Scenario: Resetting a DB auto-increment

This scenario describes a single component Job which aims at re-initializing the DB auto-increment to 1. This job has no output and is generally to be used before running a script.

⚠️ Warning:
As a prerequisite of this Job, the relevant DBMS’s ODBC driver must have been installed and the corresponding ODBC connection must have been configured.
Procedure

1. Drag and drop a **tDBSQLRow** component from the **Palette** to the design workspace.

   ![tDBSQLRow component](image)

2. Double-click **tDBSQLRow** to open its **Basic settings** view.

   ![Basic settings view](image)

3. Select **Repository** in the **Property Type** list as the ODBC connection has been configured and saved in the Repository. The follow-up fields get filled in automatically.

   For more information on storing DB connections in the Repository, see *Talend Studio User Guide*.

4. The **Schema** is built-in for this Job and it does not really matter in this example as the action is made on the table auto-increment and not on data.

5. The **Query type** is also built-in. Click on the [...] button next to the **Query** statement box to launch the SQLbuilder editor, or else type in directly in the statement box:

   ```sql
   ALTER table <TableName> auto_increment = 1
   ```

6. Press **Ctrl+S** to save the Job and **F6** to run.

   The database autoincrement is reset to 1.
tDenormalize

Denormalizes the input flow based on one column.

**tDenormalize Standard properties**

These properties are used to configure tDenormalize running in the Standard Job framework.

The Standard tDenormalize component belongs to the Processing family.

The component in this framework is generally available.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

**To denormalize**

In this table, define the parameters used to denormalize your columns.

**Column**: Select the column to denormalize.

**Delimiter**: Type in the separator you want to use to denormalize your data between double quotes.

**Merge same value**: Select this check box to merge identical values.
## Advanced settings

### tStatCatcher Statistics

Select this check box to collect the log data at component level. Note that this check box is not available in the Map/Reduce version of the component.

## Global Variables

### Global Variables

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

- **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

  A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

  To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

  For further information about variables, see *Talend Studio User Guide*.

## Usage

### Usage rule

This component can be used as intermediate step in a data flow.

### Limitation

Note that this component may change the order in the incoming Java flow.

## Scenario 1: Denormalizing on one column

This scenario illustrates a Job denormalizing one column in a delimited file.

### Denormalizing on one column

#### Procedure

1. Drop the following components: `tFileInputDelimited`, `tDenormalize`, `tLogRow` from the **Palette** to the design workspace.
2. Connect the components using **Row main** connections.
3. On the **tFileInputDelimited Component** view, set the filepath to the file to be denormalized.
4. Define the **Header**, **Row Separator** and **Field Separator** parameters.

5. The input file schema is made of two columns, **Fathers** and **Children**.

6. In the **Basic settings** of tDenormalize, define the column that contains multiple values to be grouped.

7. In this use case, the column to denormalize is **Children**.

8. Set the **Delimiter** to separate the grouped values. Beware as only one column can be denormalized.

9. Select the **Merge same value** check box, if you know that some values to be grouped are strictly identical.

10. Save your Job and press **F6** to execute it.
Results

Starting job tDenormalize at 10:50 09/03/2016.

Michael|Océane
Stéphane|Agothe; Clémence
Pierrick|Erwan; Tiphaine
Robert|Xenon
Richard|Roméo
Fabrice|Martin

Job tDenormalize ended at 10:50 09/03/2016. [exit code=0]

All values from the column Children (set as column to denormalize) are grouped by their Fathers column. Values are separated by a comma.

Scenario 2: Denormalizing on multiple columns

This scenario illustrates a Job denormalizing two columns from a delimited file.

Scenario 2: Denormalizing on multiple columns

Procedure

1. Drop the following components: tFileInputDelimited, tDenormalize, tLogRow from the Palette to the design workspace.
2. Connect all components using a Row main connection.
3. On the tFileInputDelimited Basic settings panel, set the filepath to the file to be denormalized.
4. Define the Row and Field separators, the Header and other information if required.
5. The file schema is made of four columns including: Name, FirstName, HomeTown, WorkTown.
6. In the **tDenormalize** component **Basic settings**, select the columns that contain the repetition. These are the columns which are meant to occur multiple times in the document. In this use case, **FirstName**, **HomeCity** and **WorkCity** are the columns against which the denormalization is performed.

7. Add as many line to the table as you need using the plus button. Then select the relevant columns in the drop-down list.

8. In the **Delimiter** column, define the separator between double quotes, to split concatenated values. For **FirstName** column, type in ",#", for **HomeCity**, type in ",§", and for **WorkCity**, type in ",¤".

9. Save your Job and press **F6** to execute it.

```
Starting job tDenormalize2 at 12:03 09/03/2010.
Moore|Deml|New York§Rio de Janeiro|Paris§Los Angeles
Joli|Angelina§Angelina§Angelina|Berlin§Berlin§Los Angeles§Los Angeles§Los Angeles
Pitt|Brad§Brad|Beverly Hills§Paris§Los Angeles§London
Willis|Bruce§Bruce§Bruce§Bruce|Paris§Paris§Madrid§Roma|Los Angeles§Madrid§Madrid§Dublin
Job tDenormalize2 ended at 12:03 09/03/2010. [exit code=0]
```

The result shows the denormalized values concatenated using a comma.

10. Back to the **tDenormalize** components **Basic settings**, in the To denormalize table, select the **Merge same value** check box to remove the duplicate occurrences.

11. Save your Job again and press **F6** to execute it.
Results

Starting job tDenormalize2 at 13:44 08/03/2010.

Moore|Demi|New York|Rio de Janeiro|Paris|Los Angeles
Joli|Angelica|Berlin|Los Angeles|Berlin|Los Angeles
Pitt|Brad|Beverly Hills|Paris|Los Angeles|London
Willie|Bruce|Paris|Madrid|Roma|Los Angeles|Madrid|Dublin

Job tDenormalize2 ended at 13:44 08/03/2010. [exit code=0]

This time, the console shows the results with no duplicate instances.
**tDenormalizeSortedRow**

Synthesizes sorted input flow to save memory.

`tDenormalizeSortedRow` combines in a group all input sorted rows. Distinct values of the denormalized sorted row are joined with item separators.

**tDenormalizeSortedRow Standard properties**

These properties are used to configure `tDenormalizeSortedRow` running in the Standard Job framework.

The Standard `tDenormalizeSortedRow` component belongs to the Processing family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. Click **Sync columns** to retrieve the schema from the previous component in the Job. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: You create the schema and store it locally for the relevant component. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job flowcharts. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Input rows count</strong></td>
<td>Enter the number of input rows.</td>
</tr>
<tr>
<td><strong>To denormalize</strong></td>
<td>Enter the name of the column to denormalize.</td>
</tr>
</tbody>
</table>
Advanced settings

| tStatCatcher Statistics | Select this checkbox to collect the log data at component level. |

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |
| | NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. |
| | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
| | To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. |
| | For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component handles flows of data therefore it requires input and output components. |

Scenario: Regrouping sorted rows

This Java scenario describes a four-component Job. It aims at reading a given delimited file row by row, sorting input data by sort type and order, denormalizing all input sorted rows and displaying the output on the Run log console.

- Drop the following components from the Palette onto the design workspace: tFileInputDelimited, tSortRow, tDenormalizeSortedRow, and tLogRow.
- Connect the four components using Row Main links.
- In the design workspace, select tFileInputDelimited.
- Click the Component tab to define the basic settings for tFileInputDelimited.
• Set Property Type to **Built-In**.

• Fill in a path to the processed file in the **File Name** field. The `name_list` file used in this example holds two columns, `id` and `first name`.

• If needed, define row and field separators, header and footer, and the number of processed rows.

• Set Schema to **Built in** and click the three-dot button next to Edit Schema to define the data to pass on to the next component. The schema in this example consists of two columns, `id` and `name`.

• In the design workspace, select **tSortRow**.

• Click the **Component** tab to define the basic settings for **tSortRow**.
• Set the **Schema Type** to **Built-In** and click **Sync columns** to retrieve the schema from the `tFileInputDelimited` component.

• In the **Criteria** panel, use the plus button to add a line and set the sorting parameters for the schema column to be processed. In this example we want to sort the `id` columns in ascending order.

• In the design workspace, select `tDenormalizeSortedRow`.

• Click the **Component** tab to define the basic settings for `tDenormalizeSortedRow`.

• Set the **Schema Type** to **Built-In** and click **Sync columns** to retrieve the schema from the `tSortRow` component.

• In the **Input rows count** field, enter the number of the input rows to be processed or press **Ctrl+Space** to access the context variable list and select the variable: `tFileInputDelimited_1_NB_LINE`.

• In the **To denormalize** panel, use the plus button to add a line and set the parameters to the column to be denormalize. In this example we want to denormalize the `name` column.

• In the design workspace, select `tLogRow` and click the **Component** tab to define its basic settings. For more information about `tLogRow`, see `tLogRow` on page 1988.

• Save your Job and press **F6** to execute it.
The result displayed on the console shows how the name column was denormalized.

```
Starting job denormalized at 14:42 01/09/2008.
bufferSize=100000 objects
Sorting buffer...
0 milliseconds for 9 objects to sort in memory. 21474047647 items/s
Writing ordered buffer in file...
16 milliseconds for 9 objects to write in file. 562 items/s

+-------------------------+
<p>| tLogRow_1               |</p>
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Harrison; Jerry</td>
</tr>
<tr>
<td>2</td>
<td>Ford</td>
</tr>
<tr>
<td>3</td>
<td>James; Goldman</td>
</tr>
<tr>
<td>4</td>
<td>Bill; Ford; Harry; Sue</td>
</tr>
</tbody>
</table>
+-------------------------+

Job denormalized ended at 14:42 01/09/2008. [exit code=0]
```
tDie

Triggers the tLogCatcher component for exhaustive log before killing the Job.

Both tDie and tWarn components are closely related to the tLogCatcher component. They generally make sense when used alongside a tLogCatcher in order for the log data collected to be encapsulated and passed on to the output defined.

This component throws an error and kills the job. If you simply want to throw a warning, see tWarn on page 4024.

**tDie Standard properties**

These properties are used to configure tDie running in the Standard Job framework.

The Standard tDie component belongs to the Logs & Errors family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Die message</th>
<th>Enter the message to be displayed before the Job is killed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error code</td>
<td>Enter the error code if need be, as an integer.</td>
</tr>
<tr>
<td>Priority</td>
<td>Set the level of priority, as an integer</td>
</tr>
</tbody>
</table>

**Note:**

- Note that any value greater than 255 cannot be used as an error code on Linux.

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIE_MESSAGES</td>
<td>the die message. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>DIE_CODE</td>
<td>the error code of the die message. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>DIE_PRIORITY</td>
<td>the priority level of the die message. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.
For further information about variables, see Talend Studio User Guide.

Usage

| Usage rule | This component cannot be used as a start component and it is generally used with a tLogCatcher for the log purpose. |

Related scenarios

For use cases in relation with tDie, see tLogCatcher scenarios:

- Catching messages triggered by a tWarn component on page 1982
- Catching the message triggered by a tDie component on page 1984
tDotNETInstantiate

Invokes the constructor of a .NET object that is intended for later reuse.

**tDotNETInstantiate instantiates an object in the .NET for later reuse.**

**tDotNETInstantiate Standard properties**

These properties are used to configure tDotNETInstantiate running in the Standard Job framework.

The Standard tDotNETInstantiate component belongs to the DotNET family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dll to load</strong></td>
<td>Type in the path, or browse to the DLL library containing the classe(s) of interest or enter the assembly's name to be used. For example, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089 for an OleDb assembly.</td>
</tr>
<tr>
<td><strong>Fully qualified class name(i.e. ClassLibrary1.NameSpace2.Class1)</strong></td>
<td>Enter a fully qualified name for the class of interest.</td>
</tr>
<tr>
<td><strong>Value(s) to pass to the constructor</strong></td>
<td>Click the plus button to add one or more values to be passed to the constructor for the object. Or, leave this table empty to call a default constructor for the object. The valid value(s) should be the parameters required by the class to be used.</td>
</tr>
</tbody>
</table>

**Advanced settings**

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

**Global Variables**

**Global Variables**

**INSTANCE**: the instance of a .NET object. This is an After variable and it returns an object.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.
### Usage

**Usage rule**

This component can be used as a start component in a flow or an independent subjob.

To use this component, you must first install the runtime DLLs, for example `janet-win32.dll` for Windows 32-bit version and `janet-win64.dll` for Windows 64-bit version, from the corresponding Microsoft Visual C++ Redistributable Package. This allows you to avoid errors like the `UnsatisfiedLinkError` on dependent DLL.

So ensure that the runtime and all of the other DLLs which the DLL to be called depends on are installed and their versions are consistent among one another.

**Note:**

The required DLLs can be installed in the `System32` folder or in the `bin` folder of the Java runtime to be used.

If you need to export a Job using this component to run it outside the Studio, you have to specify the runtime container of interest by setting the `-Djava.library.path` argument accordingly.

For users of Talend solutions with ESB, to run a Job using this component in ESB Runtime, you need to copy the runtime DLLs to the `%KARAF_HOME%/lib/wrapper/` directory.

### Related scenario

For a related scenario, see Scenario: Utilizing .NET in Talend on page 643.
tDotNETRow

Facilitates data transform by utilizing custom or built-in .NET classes.
tDotNETRow sends data to and from libraries and classes within .NET or other custom DLL files.

**tDotNETRow Standard properties**

These properties are used to configure tDotNETRow running in the Standard Job framework.
The Standard tDotNETRow component belongs to the DotNET family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either built-in or remotely stored in the Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in:</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository:</strong></td>
<td>Select the Repository file where properties are stored. The following fields are pre-filled in using fetched data</td>
</tr>
<tr>
<td>Use a static method</td>
<td>Select this check box to invoke a static method in .NET and this will disable Use an existing instance check box.</td>
</tr>
<tr>
<td>Propagate a data to output</td>
<td>Select this check box to propagate a transformed data to output.</td>
</tr>
<tr>
<td>Use an existing instance</td>
<td>Select this check box to reuse an existing instance of a .NET object from the Existing instance to use list.</td>
</tr>
<tr>
<td><strong>Existing instance to use:</strong></td>
<td>Select an existing instance of .NET objects created by the other .NET components from the list.</td>
</tr>
<tr>
<td>Note:</td>
<td>This check box will be disabled if you have selected Use a static method and selecting this check box will disable Dll to load, Fully qualified class name(i.e. ClassLibrary1.NameSpace2.Class1) and Value(s) to pass to the constructor.</td>
</tr>
<tr>
<td>Dll to load</td>
<td>Type in the path, or browse to the DLL library containing the class(es) of interest or enter the assembly’s name to be used. For example, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089 for an OleDb assembly.</td>
</tr>
<tr>
<td>Fully qualified class name(i.e. ClassLibrary1.NameSpace2.Class1)</td>
<td>Enter a fully qualified name for the class of interest.</td>
</tr>
<tr>
<td>Method name</td>
<td>Fill this field with the name of the method to be invoked in .NET.</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>Value(s) to pass to the constructor</td>
<td>Click the plus button to add one or more lines for values to be passed to the constructor for the object. Or, leave this table empty to call a default constructor for the object. The valid value(s) should be the parameters required by the class to be used.</td>
</tr>
<tr>
<td>Method Parameters</td>
<td>Click the plus button to add one or more lines for parameters to be passed to the method.</td>
</tr>
<tr>
<td>Output value target column</td>
<td>Select a column in the output row from the list to put value into it.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>Create a new instance at each row</th>
<th>Select this check box to create a new instance at each row that passes through the component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method doesn’t return a value</td>
<td>Select this check box to invoke a method without returning a value as a result of the processing.</td>
</tr>
<tr>
<td>Returns an instance of a .NET Object</td>
<td>Select this check box to return an instance of a .NET object as a result of a invoked method.</td>
</tr>
<tr>
<td>Store the returned value for later use</td>
<td>Select this check box to store the returned value of a method for later reuse in another tDotNETRow component.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

**Global Variables**

- **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).
Usage

Usage rule

This component is utilized to integrate with .NET objects.

To use this component, you must first install the runtime DLLs, for example Janet-win32.dll for Windows 32-bit version and Janet-win64.dll for Windows 64-bit version, from the corresponding Microsoft Visual C++ Redistributable Package. This allows you to avoid errors like the UnsatisfiedLinkError on dependent DLL.

So ensure that the runtime and all of the other DLLs which the DLL to be called depends on are installed and their versions are consistent among one another.

Note:
The required DLLs can be installed in the System32 folder or in the bin folder of the Java runtime to be used.

If you need to export a Job using this component to run it outside the Studio, you have to specify the runtime container of interest by setting the -Djava.library.path argument accordingly.

For users of Talend solutions with ESB, to run a Job using this component in ESB Runtime, you need to copy the runtime DLLs to the %KARAF_HOME%/lib/wrapper/ directory.

Scenario: Utilizing .NET in Talend

This scenario describes a three-component Job that uses a DLL library containing a class called Test1.Class1 Class and invokes a method on it that processes the value and output the result onto the console.

Prerequisites

Before replicating this scenario, you need first to build up your runtime environment.

- Create the DLL to be loaded by tDotNETInstantiate
This example class built into .NET reads as follows:

```csharp
using System;
using System.Collections.Generic;
using System.Text;

namespace Test1
{
    public class Class1
    {
        string s = null;
        public Class1(string s)
        {
            this.s = s;
        }

        public string getValue()
        {
            return "Return Value from Class1: " + s;
        }
    }
}
```

This class reads the input value and adds the text *Return Value from Class1:* in front of this value. It is compiled using the latest .NET.

- Install the runtime DLL from the latest .NET. In this scenario, we use *janet-win32.dll* on Windows 32-bit version and place it in the *System32* folder.

Thus the runtime DLL is compatible with the DLL to be loaded.

### Connecting components

#### Procedure

1. Drop the following components from the Palette to the design workspace: **tDotNETInstantiate**, **tDotNETRow** and **tLogRow**.
2. Connect **tDotNETInstantiate** to **tDotNETRow** using a **Trigger On Subjob OK** connection.
3. Connect **tDotNETRow** to **tLogRow** using a **Row Main** connection.

### Configuring tDotNETInstantiate

#### Procedure

1. Double-click **tDotNETInstantiate** to display its **Basic settings** view and define the component properties.
2. Click the three-dot button next to the **Dll to load** field and browse to the DLL file to be loaded. Alternatively, you can fill the field with an assembly. In this example, we use: "C:/Program Files/ClassLibrary1/bin/Debug/ClassLibrary1.dll"

3. Fill the **Fully qualified class name** field with a valid class name to be used. In this example, we use: "Test1.Class1"

4. Click the plus button beneath the **Value(s) to pass to the constructor** table to add a new line for the value to be passed to the constructor.
   In this example, we use: "Hello world"

**Configuring tDotNETRow**

**Procedure**

1. Double-click **tDotNETRow** to display its **Basic settings** view and define the component properties.

   2. Select **Propagate data to output** check box.
   3. Select **Use an existing instance** check box and select **tDotNETInstantiate_1** from the **Existing instance to use** list on the right.
4. Fill the **Method Name** field with a method name to be used. In this example, we use "getValue", a custom method.

5. Click the three-dot button next to **Edit schema** to add one column to the schema.

<table>
<thead>
<tr>
<th>tDotNETRow_1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Column</strong></td>
</tr>
<tr>
<td>newColumn</td>
</tr>
</tbody>
</table>

Click the plus button beneath the table to add a new column to the schema and click **OK** to save the setting.

6. Select **newColumn** from the **Output value target column** list.

**Configuring tLogRow**

**Procedure**

1. Double-click **tLogRow** to display its **Basic settings** view and define the component properties.

2. Click **Sync columns** button to retrieve the schema defined in the preceding component.

3. Select **Table** in the **Mode** area.

**Results**

Save your Job and press **F6** to execute it.
From the result, you can read that the text `Return Value from Class1` is added in front of the retrieved value `Hello world`. 
tDropboxConnection

Creates a Dropbox connection to a given account that the other Dropbox components can reuse.

**tDropboxConnection Standard properties**

These properties are used to configure tDropboxConnection running in the Standard Job framework. The Standard tDropboxConnection component belongs to the Cloud family. The component in this framework is generally available.

**Basic settings**

| Access Token | Enter the access token required by the Dropbox account you need to connect to. This access token allows the Studio to make Dropbox API calls for that Dropbox account. Note that a Dropbox App should have been created under that account before generating the access token. For further information about a Dropbox access token, see https://www.dropbox.com/developers/blog/94/generate-an-access-token-for-your-own-account. |
| Use HTTP Proxy | If you are using a proxy, select this check box and enter the host and port information of that proxy in the corresponding fields that are displayed. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

**Usage**

| Usage rule | This component is used standalone as a Subjob to create the Dropbox connection to be used. In a Job design, it is often connected to the other Dropbox components using the Trigger links such as On Subjob Ok link. |

**Related scenario**

See Scenario: Uploading files to Dropbox on page 657
tDropboxDelete

Removes a given folder or file from Dropbox.

**tDropboxDelete Standard properties**

These properties are used to configure tDropboxDelete running in the Standard Job framework. The Standard tDropboxDelete component belongs to the Cloud family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Existing Connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>Access Token</td>
<td>Enter the access token required by the Dropbox account you need to connect to. This access token allows the Studio to make Dropbox API calls for that Dropbox account. Note that a Dropbox App should have been created under that account before generating the access token. For further information about a Dropbox access token, see <a href="https://www.dropbox.com/developers/blog/94/generate-an-access-token-for-your-own-account">https://www.dropbox.com/developers/blog/94/generate-an-access-token-for-your-own-account</a>.</td>
</tr>
<tr>
<td>Use HTTP Proxy</td>
<td>If you are using a proxy, select this check box and enter the host and port information of that proxy in the corresponding fields that are displayed.</td>
</tr>
<tr>
<td>Path</td>
<td>Enter the path on Dropbox pointing to the folder or the file you need to remove. Note that the path string should start with a slash (/). It is the root folder of the Dropbox App for which you are using the current access token.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>
**Usage**

| Usage rule | This component is usually used standalone in a Subjob to remove data from Dropbox. |

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tDropboxGet

Downloads a selected file from a Dropbox account to a specified local directory.

**tDropboxGet Standard properties**

These properties are used to configure tDropboxGet running in the Standard Job framework.

The Standard tDropboxGet component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use Existing Connection</th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
</table>
| **Access Token**        | Enter the access token required by the Dropbox account you need to connect to. This access token allows the Studio to make Dropbox API calls for that Dropbox account.  
Note that a Dropbox App should have been created under that account before generating the access token.  
For further information about a Dropbox access token, see [https://www.dropbox.com/developers/blog/94/generate-an-access-token-for-your-own-account](https://www.dropbox.com/developers/blog/94/generate-an-access-token-for-your-own-account). |
| **Use HTTP Proxy**      | If you are using a proxy, select this check box and enter the host and port information of that proxy in the corresponding fields that are displayed. |
| **Path**                | Enter the path on Dropbox pointing to the file you need to download.  
Note that the path string should start with a slash (/). It is the root folder of the Dropbox App for which you are using the current access token. |
| **Save As File**        | Select this check box to display the **File** field and browse to, or enter the local directory where you want to store the downloaded file. The existing file, if any, is replaced. |
| **Schema** and **Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component.  
The schema of this component is read-only. You can click the **Edit schema** button next to **Edit schema** to view the predefined schema that contains the following two columns:  
  - **fileName**: the name of the downloaded file.  
  - **content**: the content of the downloaded file. |
Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

Global Variables

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

Usage

| Usage rule | This component can be used alone or along with other components via the Iterate link or a trigger link such as On Subjob OK. |

Related scenarios

No scenario is available for the Standard version of this component yet.
tDropboxList

Lists the files stored in a specified directory on Dropbox.

**tDropboxList Standard properties**

These properties are used to configure tDropboxList running in the Standard Job framework.

The Standard tDropboxList component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Use Existing Connection</strong></th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
</table>
| **Access Token**            | Enter the access token required by the Dropbox account you need to connect to. This access token allows the Studio to make Dropbox API calls for that Dropbox account.  
Note that a Dropbox App should have been created under that account before generating the access token.  
For further information about a Dropbox access token, see https://www.dropbox.com/developers/blog/94/generate-an-access-token-for-your-own-account. |
| **Use HTTP Proxy**          | If you are using a proxy, select this check box and enter the host and port information of that proxy in the corresponding fields that are displayed. |
| **Path**                    | Enter the path pointing to the folder you need to list the files from, or enter the path pointing to the exact file you need to read.  
Note that the path string should start with a slash (/). It is the root folder of the Dropbox App for which you are using the current access token. |
| **List Type**               | Select the type of data you need to list from the specified path. |
| **Include subdirectories**  | Select this check box to list files from any existing subfolders in addition to the files in the directory defined in the Path field. |

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |
Global Variables

<table>
<thead>
<tr>
<th>NAME</th>
<th>The name of the remote file being processed. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATH</td>
<td>The path to the folder or the file being processed on Dropbox. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>LAST_MODIFIED</td>
<td>The timestamp of the last modification of the file being processed. This is a Flow variable and it returns a long.</td>
</tr>
<tr>
<td>SIZE</td>
<td>The volume of the file being processed. This is a Flow variable and it returns a long.</td>
</tr>
<tr>
<td>IS_FILE</td>
<td>The boolean result of the file listing. This is a Flow variable and it returns a boolean. The result Yes indicates that the listed data is of the type File; otherwise, the type is Folder.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule                      | This component is typically used standalone. |

Related scenarios

No scenario is available for the Standard version of this component yet.
tDropboxPut

Uploads data to Dropbox from either a local file or a given data flow.

tDropboxPut Standard properties

These properties are used to configure tDropboxPut running in the Standard Job framework.

The Standard tDropboxPut component belongs to the Cloud family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Use Existing Connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Token</td>
<td>Enter the access token required by the Dropbox account you need to connect to. This access token allows the Studio to make Dropbox API calls for that Dropbox account. Note that a Dropbox App should have been created under that account before generating the access token. For further information about a Dropbox access token, see <a href="https://www.dropbox.com/developers/blog/94/generate-an-access-token-for-your-own-account">https://www.dropbox.com/developers/blog/94/generate-an-access-token-for-your-own-account</a>.</td>
</tr>
<tr>
<td>Use HTTP Proxy</td>
<td>If you are using a proxy, select this check box and enter the host and port information of that proxy in the corresponding fields that are displayed.</td>
</tr>
<tr>
<td>Path (File Only)</td>
<td>Enter the path pointing to the file you need to write contents in. This file will be created on the fly if it does not exist. Note that the path string should start with a slash (/). It is the root folder of the Dropbox App for which you are using the current access token.</td>
</tr>
</tbody>
</table>
| Upload Mode             | Select upload mode to be used:  
  - **Rename if Existing**: the uploaded file is automatically renamed. For example, a file named test.txt might be renamed to test (1).txt.  
  - **Replace if Existing**: the uploaded file replaces the existing one.  
  - **Update specified Revision**: the file you are uploading is used to update a specific revision of that file. If the revision you specify is the latest revision, then the existing file on Dropbox is replaced; if it is an older revision, the file you are uploading is renamed to indicate that a conflict is encountered; if the revision does not exist, an error is returned. |
<table>
<thead>
<tr>
<th><strong>Upload Incoming content as File</strong></th>
<th>Select this radio button to read data directly from the input flow of the preceding component and write the data into the file specified in the Path field.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields. Note that the schema of this component is read-only with a single column named <strong>content</strong> and it receives data from the <strong>content</strong> column of its input schema only. This means that you must use a <strong>content</strong> column in the input data flow to carry the data to be uploaded. This type of column is typically provided by the <strong>tFileInputRaw</strong> component. For further information, see <strong>tFileInputRaw</strong> on page 1081. The Schema field is not available when you have selected the Expose as OutputStream or the Upload local file radio button.</td>
</tr>
<tr>
<td><strong>Upload local file</strong></td>
<td>Select this radio button to upload a locally stored file to Dropbox. In the File field that is displayed, you need to enter the path or browse to this file.</td>
</tr>
<tr>
<td><strong>Expose as OutputStream</strong></td>
<td>Select this check box to expose the output stream of this component as a variable named <strong>OUTPUTSTREAM</strong> so that the other components can reuse this variable to write the contents to be uploaded into the exposed output stream. For example, you can use the Use output stream feature of the <strong>tFileOutputDelimited</strong> component to feed a given <strong>tDropboxPut</strong>'s exposed output stream. For further information, see <strong>tFileOutputDelimited</strong> on page 1110.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| **ERROR_MESSAGE** | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

**Usage**

| **Usage rule** | This component is used either standalone in a Subjob to directly upload a local file to Dropbox or as an end component of a Job flow to upload given data being handled in this flow. |
Scenario: Uploading files to Dropbox

In this scenario, a six-component Job consisting of three Subjobs is created to write data onto Dropbox using different upload modes.

Before replicating this scenario, you need to create a Dropbox App under the Dropbox account to be used. In this scenario, the Dropbox App to be used is named to talenddrop and thus the root folder in which files are uploaded is talenddrop, too. In addition, the access token to this folder has been generated from the App console provided by Dropbox.

For further information about a Dropbox App, see https://www.dropbox.com/developers/apps/.

Linking the components

Procedure

1. In the Integration perspective of the Studio, create an empty Job from the Job Designs node in the Repository tree view.
   For further information about how to create a Job, see Talend Studio User Guide.
2. In the workspace, enter the name of the component to be used and select this component from the list that appears. In this scenario, the components are tDropboxConnection, tFixedFlowInput, tFileOutputDelimited, tFileInputRaw and two tDropboxPut components.
   The tFixedFlowInput component generates some data to be uploaded to Dropbox in this scenario. In the real-world case, you can use other components such as tMysqlInput or tMap in the place of tFixedFlowInput to design a sophisticated process to prepare your data to be handled.
3. Connect tFixedFlowInput to tFileOutputDelimited using the Row > Main link.
4. Do the same to connect tFileOutputDelimited to one of the two tDropboxPut components and connect tFileInputRaw to the other tDropboxPut component.
5. Connect tDropboxConnection to tFixedFlowInput using the Trigger > On Subjob Ok link. Then connect tFixedFlowInput to tFileInputRaw using the same type of link.
## Connecting to Dropbox

### Procedure

1. Double-click `tDropboxConnection` to open its **Component** view.

   ![tDropboxConnection](image)

   In the **Access token** field, paste the token that you have generated via the App console of Dropbox for accessing the Dropbox App folder to be used.

2. In the **Access token** field, paste the token that you have generated via the App console of Dropbox for accessing the Dropbox App folder to be used.

## Generating the output stream

### Defining the input data

### Procedure

1. Double-click `tFixedFlowInput` to open its **Component** view.

   ![tFixedFlowInput](image)

   In this scenario, only three rows of sample data are created to indicate three countries and their calling codes.

   ```
   33; France
   86; China
   81; Japan
   ```

2. Click the `[...]` button next to **Edit schema** to open the schema editor.
3. Click the [+] button twice to add two rows and in the Column column, rename them to code and country, respectively.

4. Click OK to validate these changes and accept the propagation prompted by the pop-up dialog box.

5. In the Mode area, select the Use Inline Table radio button. The code and the country column have been automatically created in this table.

6. Enter the sample data mentioned above in this table.

Defining the output stream

Procedure

1. Double-click tFileOutputDelimited to open its Component view.

2. Select the Use output stream check box to write the data to be outputted into a given output stream.

3. In the Output stream field, enter the code to define the output stream you need to write data in. In this scenario, it is the output stream of the tDropboxPut_1 component linked with the current component. Thus the code used to write the data reads as follows:

   ```java
   ((java.io.OutputStream)globalMap.get("tDropboxPut_1_OUTPUTSTREAM"))
   ```

   Note that in this example code, the tDropboxPut component has the number 1 as its affix, which represents its component ID distributed automatically within this Job. If the tDropboxPut component you are using has a different ID, you need to adapt the code to that ID number.

4. Click Edit schema to verify that the schema of this component is identical with that of the preceding tFixedFlowInput component. If not so, click the Sync columns button to make both of the schemas identical.
Exposing the tDropboxPut output stream

Procedure

1. Double-click the tDropboxPut component linked with tFileOutputDelimited to open its Component view.

   ![Component View of tDropboxPut](image1)

   2. Select the Use existing connection check box to reuse the connection created by tDropboxConnection.

   3. In the Path field, enter the path pointing to the file you need to write data in, with a slash (/) at the beginning of the path. For example, enter /calling_code.csv.

   4. In the Upload mode area, select the Rename if Existing radio button.

   5. Select the Expose As OutputStream radio button to expose the output stream of this component so that the other component, tFileOutputDelimited in this scenario, can write data in the stream.

Defining the media data to be uploaded

Procedure

1. Double-click tFileInputRaw to open its Component view.

   ![Component View of tFileInputRaw](image2)

   This component is used to read a picture named esb_architecture.png into the data flow. In the real-world practice, this file can be of many other formats, such as pdf, xls, ppt or mp3.

   2. In the Filename field, enter the path or browse to the file you need to upload.

   3. In the Mode area, select the Read the file as a bytes array radio button.
Uploading the incoming contents

Procedure

1. Double-click the tDropboxPut component linked with tFileInputRaw to open its Component view.

2. Select the Use existing connection check box to reuse the connection created by tDropboxConnection.

3. In the Path field, enter the path pointing to the file you need to write data in, with a slash (/) at the beginning of the path. For example, enter /architecture.png.

4. In the Upload mode area, select Rename if existing.

5. Select the Upload incoming content as file radio button. This displays the Edit schema button to allow you to view the read-only schema of this component.

Executing the Job

Then you can press F6 to run this Job.

Once done, check the uploaded files in the Dropbox App folder of your Dropbox, in this scenario, the talenddrop folder.
**tDTDValidator**

Helps at controlling data and structure quality of the file to be processed
Validates the XML input file against a DTD file and sends the validation log to the defined output.

**tDTDValidator Standard properties**

These properties are used to configure tDTDValidator running in the Standard Job framework.
The Standard tDTDValidator component belongs to the XML family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema of this component is read-only. It contains standard information regarding the file validation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTD file</td>
<td>Filepath to the reference DTD file.</td>
</tr>
<tr>
<td>XML file</td>
<td>Filepath to the XML file to be validated.</td>
</tr>
<tr>
<td>If XML is valid, display</td>
<td>If XML is invalid, display</td>
</tr>
<tr>
<td>If XML is valid, display</td>
<td>Type in a message to be displayed in the Run console based on the result of the comparison.</td>
</tr>
<tr>
<td>Print to console</td>
<td>Select this check box to display the validation message.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics                       | Select this check box to gather the processing metadata at the Job level as well as at each component level.                                                                                     |

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
<td></td>
</tr>
<tr>
<td><strong>DIFFERENCE</strong>: the result of the validation. This is a Flow variable and it returns a string.</td>
<td></td>
</tr>
<tr>
<td><strong>VALID</strong>: the validation result. This is a Flow variable and it returns a boolean.</td>
<td></td>
</tr>
<tr>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
<td></td>
</tr>
</tbody>
</table>
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

This component can be used as standalone component but it is usually linked to an output component to gather the log data.

Scenario: Validating XML files

This scenario describes a Job that validates the specified type of files from a folder, displays the validation result on the Run tab console, and outputs the log information for the invalid files into a delimited file.

Validating XML files

Procedure

1. Drop the following components from the Palette to the design workspace: tFileList, tDTDValidator, tMap, tFileOutputDelimited.
2. Connect the tFileList to the tDTDValidator with an Iterate link and the remaining component using a main row.
3. Set the tFileList component properties, to fetch an XML file from a folder.

   Directory: D:/Input
   Filelist Type: Files
   Includes subdirectories: No
   Case Sensitive: No
   Generate Error if no file found: No
   Use Glob Expressions as Filemask (Unchecked means Perl5 Regex Expressions)

   Filemask:
   *.xml

   Click the plus button to add a filemask line and enter the filemask: *.xml. Remember Java code requires double quotes.
Set the path of the XML files to be verified.
Select No from the Case Sensitive drop-down list.

4. In the tDTDValidate Component view, the schema is read-only as it contains standard log information related to the validation process.

5. Click in the XML file field, press Ctrl+Space bar to access the variable list, and double-click the current filepath global variable: tFileList.CURRENT_FILEPATH.

6. In the various messages to display in the Run tab console, use the jobName variable to recall the job name tag. Recall the filename using the relevant global variable: ((String)globalMap.get("tFileList_1_CURRENT_FILE"))). Remember Java code requires double quotes.
Select the Print to Console check box.

7. In the tMap component, drag and drop the information data from the standard schema that you want to pass on to the output file.

8. Once the Output schema is defined as required, add a filter condition to only select the log information data when the XML file is invalid.

Follow the best practice by typing first the wanted value for the variable, then the operator based on the type of data filtered then the variable that should meet the requirement. In this case: 0 == row1.validate.

9. Then connect (if not already done) the tMap to the tFileOutputDelimited component using a Row > Main connection. Name it as relevant, in this example: log_errorsOnly.

10. In the tFileOutputDelimited Basic settings, define the destination filepath, the field delimiters and the encoding.

11. Save your Job and press F6 to run it.
On the Run console the messages defined display for each of the files. At the same time the output file is filled with the log data for invalid files.
tDynamoDBInput

Retrieves data from an Amazon DynamoDB table and sends them to the component that follows for transformation.

**tDynamoDBInput Standard properties**

These properties are used to configure tDynamoDBInput running in the Standard Job framework.

The Standard tDynamoDBInput component belongs to the Big Data family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

### Basic settings

| **Access Key** | Enter the access key ID that uniquely identifies an AWS Account. For further information about how to get your Access Key and Secret Key, see Getting Your AWS Access Keys. |
| **Secret Key** | Enter the secret access key, constituting the security credentials in combination with the access Key. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Inherit credentials from AWS role** | Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances. |
| **Assume role** | Select this check box and specify the values for the following parameters used to create a new assumed role session.  
   - **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.  
   - **Role session name**: an identifier for the assumed role session.  
   - **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.  
   For more information on assuming roles, see AssumeRole. |
<p>| <strong>Use End Point</strong> | Select this check box and in the Server Url field displayed, specify the Web service URL of the DynamoDB database service. |</p>
<table>
<thead>
<tr>
<th>Region</th>
<th>Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (e.g. &quot;us-east-1&quot;) in the list. For more information about the AWS Region, see Regions and Endpoints.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Select the operation to be performed from the dropdown list, either Query or Scan. For more information, see Query and Scan Operations in DynamoDB.</td>
</tr>
</tbody>
</table>
| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
  • **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.  
  • **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. |
| Table Name      | Specify the name of the table to be queried or scanned.                                                                                                                                                 |
| Use advanced key condition expression | Select this check box and in the Advanced key condition expression field displayed, specify the key condition expressions used to determine the items to be read from the table or index. |
| Key condition expression | Specify the key condition expressions used to determine the items to be read. Click the [+] button to add as many rows as needed, each row for a key condition expression, and set the following attributes for each expression:  
  • **Key Column**: Enter the name of the key column.  
  • **Function**: Select the function for the key condition expression.  
  • **Value1**: Specify the value used in the key condition expression. |
| **tDynamoDBInput** | **Value2**: Specify the second value used in the key condition expression if needed, depending on the function you selected.  
Note that only the items that meet all the key conditions defined in this table can be returned.  
This table is not available when the **Use advanced key condition expression** check box is selected. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use filter expression</strong></td>
<td>Select this check box to use the filter expression for the query or scan operation.</td>
</tr>
</tbody>
</table>
| **Use advanced filter expression** | Select this check box and in the Advanced filter expression field displayed, specify the filter expressions used to refine the data after it is queried or scanned and before it is returned to you.  
This check box is available when the **Use filter expression** check box is selected. |
| **Filter expression** | Specify the filter expressions used to refine the results returned to you. Click the [+] button to add as many rows as needed, each row for a filter expression, and set the following attributes for each expression:  
• **Column**: Enter the name of the column used to refine the results.  
• **Function**: Select the function for the filter expression.  
• **Value1**: Specify the value used in the filter expression.  
• **Value2**: Specify the second value used in the filter expression if needed, depending on the function you selected.  
Note that only the items that meet all the filter conditions defined in this table can be returned.  
This table is available when the **Use filter expression** check box is selected and the **Use advanced filter expression** check box is cleared. |
| **Value mapping** | Specify the placeholders for the expression attribute values.  
• **value**: Enter the expression attribute value.  
• **placeholder**: Specify the placeholder for the corresponding value.  
For more information, see Expression Attribute Values. |
| **Name mapping** | Specify the placeholders for the attribute names that conflict with the DynamoDB reserved words.  
• **name**: Enter the name of the attribute that conflicts with a DynamoDB reserved word.  
• **placeholder**: Specify the placeholder for the corresponding attribute name.  
For more information, see Expression Attribute Names. |
Advanced settings

<table>
<thead>
<tr>
<th>STS Endpoint</th>
<th>Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from. This check box is available only when the Assume role check box is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

Global Variables

| Global Variables | NB_LINE: the number of rows processed. This is an After variable and it returns an integer.  
QUERY: the query statement being processed. This is a Flow variable and it returns a string.  
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component is usually used as a start component of a Job or Subjob and it always needs an output link. |

Related scenarios

No scenario is available for the Standard version of this component yet.
tDynamoDBOutput

Writes data into an Amazon DynamoDB table.
tDynamoDBOutput creates, updates or deletes data in an Amazon DynamoDB table.

**tDynamoDBOutput Standard properties**

These properties are used to configure tDynamoDBOutput running in the Standard Job framework.
The Standard tDynamoDBOutput component belongs to the Big Data family.
The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Access Key</strong></th>
<th>Enter the access key ID that uniquely identifies an AWS Account. For further information about how to get your Access Key and Secret Key, see <a href="#">Getting Your AWS Access Keys</a>.</th>
</tr>
</thead>
</table>
| **Secret Key** | Enter the secret access key, constituting the security credentials in combination with the access Key.  
To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| **Inherit credentials from AWS role** | Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances](#). |
| **Assume role** | Select this check box and specify the values for the following parameters used to create a new assumed role session.  
- **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.  
- **Role session name**: an identifier for the assumed role session.  
- **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.  
For more information on assuming roles, see [AssumeRole](#). |
<table>
<thead>
<tr>
<th><strong>Use End Point</strong></th>
<th>Select this check box and in the <strong>Server Url</strong> field displayed, specify the Web service URL of the DynamoDB database service.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Region</strong></td>
<td>Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (e.g. &quot;us-east-1&quot;) in the list. For more information about the AWS Region, see <strong>Regions and Endpoints</strong>.</td>
</tr>
</tbody>
</table>
| **Action on table** | Select an operation to be performed on the table defined.  
- **Default**: No operation is carried out.  
- **Drop and create table**: The table is removed and created again.  
- **Create table**: The table does not exist and gets created.  
- **Create table if does not exist**: The table is created if it does not exist.  
- **Drop table if exist and create**: The table is removed if it already exists and created again. |
| **Action on data** | On the data of the table defined, you can perform one of the following operations:  
- **Insert**: Insert new items from the input flow.  
- **Update**: Update existing items according to the input flow.  
- **Delete**: Remove existing items according to the input flow. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.  
- **Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.  
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.  
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate
the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Table Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the name of table to be written.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Partition Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the partition key of the specified table.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sort Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the sorted key of the specified table.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>STS Endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from. This check box is available only when the Assume role check box is selected.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Read Capacity Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the number of read capacity units. For more information, see Amazon DynamoDB Provisioned Throughput.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Write Capacity Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the number of write capacity units. For more information, see Amazon DynamoDB Provisioned Throughput.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer. <strong>QUERY</strong>: the query statement being processed. This is a Flow variable and it returns a string. <strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
Usage

| Usage rule | This component is usually used as an end component of a Job or Subjob and it always needs an input link. |

Related scenarios

No scenario is available for the Standard version of this component yet.
tEDIFACTtoXML

Transforms an EDIFACT message file into the XML format for better readability to users and compatibility with processing tools.

This component reads a United Nations/Electronic Data Interchange For Administration, Commerce and Transport (UN/EDIFACT) message and transforms it into the XML format according to the EDIFACT version and the EDIFACT family.

**tEDIFACTtoXML Standard properties**

These properties are used to configure tEDIFACTtoXML running in the Standard Job framework.

The Standard tEDIFACTtoXML component belongs to the XML family.

The component in this framework is generally available.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema of this component is fixed and read-only, with only one column: *document*. |
| **EDI filename** | Filepath to the EDIFACT message file to be transformed. |
| **EDI version** | Select the EDIFACT version of the input file. |
| **Ignore new line** | Select this check box to skip carriage returns in the input file. |
| **Die on error** | Select this check box to stop Job execution when an error is encountered. By default, this check box is cleared, and therefore illegal rows are skipped and the process is completed for the error free rows. |

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

| **Global Variables** | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

| Usage rule | This component is usually linked to an output component to gather the transformation result. |

Scenario: From EDIFACT to XML

This scenario describes a simple Job that reads a UN/EDIFACT Customs Cargo (CUSCAR) message file and saves it as an XML file.

Adding and linking the components

Procedure

1. Drop the tEDIFACTtoXML component and the tFileOutputXML component from the Palette to the design workspace.
2. Connect the tEDIFACTtoXML component and the tFileOutputXML component using a Row > Main connection.

Results

Configuring the components

Procedure

1. Double-click the tEDIFACTtoXML component to show its Basic settings view.

   | Basic settings |  
   |-------|-------|
   | EDI filename | "D:\Input\EDIFACT\99a_cuscar.edi" |
   | EDI version | D99A |

2. Fill the EDI filename field with the full path to the input EDIFACT message file.
   In this use case, the input file is 99a_cuscar.edi.
3. From **EDI version** list, select the EDIFACT version of the input file, *D99A* in this use case.

4. Select the **Ignore new line** check box to skip the carriage return characters in the input file during the transformation.

5. Leave the other parameters as they are.

6. Double-click the **tFileOutputXML** component to show its **Basic settings** view.

   ![tFileOutputXML component](image)

   - **File Name**: [D:\Output\EDIFACT.XML\99a_cuscar.xml]
   - **Row tag**: "row"

   Leave the other parameters as they are.

7. Fill the **File Name** field with the full path to the output XML file you want to generate.

   In this use case, the output XML is *99a_cuscar.xml*.

8. Leave the other parameters as they are.

### Saving and executing the Job

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to run the Job.

**Results**

The input EDIFACT CUSCAR message file is transformed into the XML format and the output XML file is generated as defined.
<document><unEdifact>
  <UNB>
    <syntaxIdentifier>
      <id>UNOA</id>
      <versionNum>2</versionNum>
    </syntaxIdentifier>
    <sender>
      <id>SENDER</id>
    </sender>
    <recipient>
      <id>XXXXXXXXLCTA</id>
    </recipient>
    <dateDateTime>
      <date>100421</date>
      <time>0437</time>
    </dateDateTime>
    <controlRef>1918</controlRef>
  </UNB>
</unEdifact>
tELTGreenplumInput

Adds as many Input tables as required for the most complicated Insert statement.

The three ELT Greenplum components are closely related, in terms of their operating conditions. These components should be used to handle Greenplum DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

Provides the table schema to be used for the SQL statement to execute.

tELTGreenplumInput Standard properties

These properties are used to configure tELTGreenplumInput running in the Standard Job framework. The Standard tELTGreenplumInput component belongs to the ELT family.

The component in this framework is generally available.

Basic settings

| **Schema** and **Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:
| | • **View schema**: choose this option to view the schema only.
| | • **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
| | • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.
| **Built-in**: The schema is created and stored locally for this component only. Related topic: see **Talend Studio User Guide**.
| **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see **Talend Studio User Guide**.
| **Default Table Name** | Type in the default table name.
| **Default Schema Name** | Type in the default schema name. |
**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  

For further information about variables, see *Talend Studio User Guide*. |

**Usage**

| Usage rule | **tELTGreenplumInput** is to be used along with the **tELTGreenplumMap**. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name  

**Note**:  
Note that the ELT components do not handle actual data flow but only schema information. |

**Related scenarios**

- **Scenario: Mapping data using a simple implicit join** on page 682  
- **Scenario 1: Aggregating table columns and filtering** on page 744  
- **Scenario 2: ELT using an Alias table** on page 748  
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTGreenplumMap

Uses the tables provided as input to feed the parameter in the built statement. The statement can include inner or outer joins to be implemented between tables or between one table and its aliases.

The three ELT Greenplum components are closely related, in terms of their operating conditions. These components should be used to handle Greenplum DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

Helps you to build the SQL statement graphically, using the table provided as input.

**tELTGreenplumMap Standard properties**

These properties are used to configure tELTGreenplumMap running in the Standard Job framework.

The Standard tELTGreenplumMap component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td></td>
<td>For an example about how to share a database connection across Job levels, see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ELT Greenplum Map Editor</th>
<th>The ELT Map editor allows you to define the output schema and make a graphical build of the SQL statement to be executed. The column names of schema can be different from the column names in the database.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Style link</th>
<th>Select the way in which links are displayed.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Auto:</strong></td>
<td>By default, the links between the input and output schemas and the Web service parameters are in the form of curves.</td>
</tr>
<tr>
<td><strong>Bezier curve:</strong></td>
<td>Links between the schema and the Web service parameters are in the form of curve.</td>
</tr>
</tbody>
</table>
**Line**: Links between the schema and the Web service parameters are in the form of straight lines. This option slightly optimizes performance.

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the Repository file where Properties are stored. The following fields are pre-filled in using fetched data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Additional JDBC parameters</th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the Use an existing connection check box in the Basic settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</th>
</tr>
</thead>
</table>
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tELTGreenplumMap is used along with tELTGreenplumInput and tELTGreenplumOutput. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note: Note that the ELT components do not handle actual data flow but only schema information.</td>
</tr>
</tbody>
</table>

| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

Scenario: Mapping data using a simple implicit join

In this scenario, a tELTGreenplumMap component is deployed to retrieve the data from the source table employee_by_statecode, compares its statecode column against the table statecode, and then maps the desired columns from the two tables to the output table employee_by_state.

Before the Job execution, the three tables, employee_by_statecode, statecode and employee_by_state look like:

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>statecode</th>
<th>state</th>
<th>statecode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Andy</td>
<td>86</td>
<td>China</td>
<td>66</td>
</tr>
<tr>
<td>3</td>
<td>Mike</td>
<td>39</td>
<td>Mexico</td>
<td>52</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>52</td>
<td>Italy</td>
<td>39</td>
</tr>
</tbody>
</table>
Dropping components

Procedure

1. Drop tGreenplumConnection, tELTGreenplumInput (two), tELTGreenplumMap, tELTGreenplumOutput, tGreenplumCommit, tGreenplumInput and tLogRow from the Palette onto the workspace.

2. Rename tGreenplumConnection as connect_to_greenplum_host, two tELTGreenplumInput components as employee+statecode and statecode, tELTGreenplumMap as match+map, tELTGreenplumOutput as map_data_output, tGreenplumCommit as commit_to_host, tGreenplumInput as read_map_output_table and tLogRow as show_map_data.

3. Link tGreenplumConnection to tELTGreenplumMap using an OnSubjobOk trigger.
   Link tELTGreenplumMap to tGreenplumCommit using an OnSubjobOk trigger.
   Link tGreenplumCommit to tGreenplumInput using an OnSubjobOk trigger.

4. Link tGreenplumInput to tLogRow using a Row > Main connection.
   The two tELTGreenplumInput components and tELTGreenplumOutput will be linked to tELTGreenplumMap later once the relevant tables have been defined.

Configuring the components

Procedure

1. Double-click tGreenplumConnection to open its Basic settings view in the Component tab.

   ![Configuration screen](image)

   In the **Host** and **Port** fields, enter the context variables for the Greenplum server.
   In the **Database** field, enter the context variable for the Greenplum database.
In the **Username** and **Password** fields, enter the context variables for the authentication credentials.

For more information on context variables, see *Talend Studio User Guide*.

2. Double-click **employee+statecode** to open its **Basic settings** view in the **Component** tab.

   ![Schema of employee+statecode](image)

   In the **Default table name** field, enter the name of the source table, namely `employee_by_statecode`.

   Click the [...] button next to the **Edit schema** field to open the schema editor.

   ![Add columns](image)

   Click the [+] button to add three columns, namely `id`, `name` and `statecode`, with the data type as `INT4`, `VARCHAR`, and `INT4` respectively.

   Click **OK** to close the schema editor.

   Link **employee+statecode** to **tELTGreenplumMap** using the output `employee_by_statecode`.

3. Double-click **statecode** to open its **Basic settings** view in the **Component** tab.

   ![Schema of statecode](image)

   In the **Default table name** field, enter the name of the lookup table, namely `statecode`.

4. Click the [...] button next to the **Edit schema** field to open the schema editor.
Click the [+] button to add two columns, namely state and statecode, with the data type as VARCHAR and INT4 respectively.

Click OK to close the schema editor.

Link statecode to tELTGreenplumMap using the output statecode.

5. Click tELTGreenplumMap to open its Basic settings view in the Component tab.

Select the Use an existing connection check box.

6. Click the [...] button next to the ELT Greenplum Map Editor field to open the map editor.
7. Click the [+ button on the upper left corner to open the table selection box.

Select tables `employee_by_statecode` and `statecode` in sequence and click Ok. The tables appear on the left panel of the editor.

8. On the upper right corner, click the [+ button to add an output table, namely `employee_by_state`. Click Ok to close the map editor.

9. Double-click `tELTGreenplumOutput` to open its Basic settings view in the Component tab.
In the **Default table name** field, enter the name of the output table, namely *employee_by_state*.

10. Click the [...] button next to the **Edit schema** field to open the schema editor.

Click the [+] button to add three columns, namely *id*, *name* and *state*, with the data type as **INT4**, **VARCHAR**, and **VARCHAR** respectively.

Click **OK** to close the schema editor.

Link **tELTGreenplumMap** to **tELTGreenplumOutput** using the table output *employee_by_state*.

Click **OK** on the pop-up window below to retrieve the schema of **tELTGreenplumOutput**.

Now the map editor’s output table *employee_by_state* shares the same schema as that of **tELTGreenplumOutput**.

11. Double-click **tELTGreenplumMap** to open the map editor.

Drop the column *statecode* from table *employee_by_statecode* to its counterpart of the table *statecode*, looking for the records in the two tables that have the same *statecode* values.

Drop the columns *id* and *name* from table *employee_by_statecode* as well as the column *statecode* from table *statecode* to their counterparts in the output table *employee_by_state*. 
Click **Ok** to close the map editor.

**12.** Double-click **tGreenplumInput** to open its **Basic settings** view in the **Component** tab.

Select the **Use an existing connection** check box.

In the **Table name** field, enter the name of the source table, namely **employee_by_state**.

In the **Query** field, enter the query statement, namely "SELECT * FROM "employee_by_state"".

**13.** Double-click **tLogRow** to open its **Basic settings** view in the **Component** tab.

In the **Mode** area, select **Table (print values in cells of a table)** for a better display.

**Executing the Job**

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to run the Job.

```plaintext
--- 3 rows inserted.
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>John</td>
<td>Mexico</td>
</tr>
<tr>
<td>1</td>
<td>Andy</td>
<td>China</td>
</tr>
<tr>
<td>3</td>
<td>Mike</td>
<td>Italy</td>
</tr>
</tbody>
</table>
```

As shown above, the desired employee records have been written to the table **employee_by_state**, presenting clearer geographical information about the employees.
Related scenarios

- **Scenario 1:** Aggregating table columns and filtering on page 744
- **Scenario 2:** ELT using an Alias table on page 748
- **Scenario:** Mapping data using a subquery on page 800, a related scenario using subquery
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTGreenplumOutput

Executes the SQL Insert, Update and Delete statement to the Greenplum database

The three ELT Greenplum components are closely related, in terms of their operating conditions. These components should be used to handle Greenplum DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

Carries out the action on the table specified and inserts the data according to the output schema defined the ELT Mapper.

tELTGreenplumOutput Standard properties

These properties are used to configure tELTGreenplumOutput running in the Standard Job framework. The Standard tELTGreenplumOutput component belongs to the ELT family.
The component in this framework is generally available.

Basic settings

| Action on data | On the data of the table defined, you can perform the following operation:  
|                | Insert: Adds new entries to the table.  
|                | Update: Updates entries in the table.  
|                | Delete: Deletes the entries which correspond to the entry flow. |
| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
|                | Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
|                | • View schema: choose this option to view the schema only.  
|                | • Change to built-in property: choose this option to change the schema to Built-in for local changes.  
|                | • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the Repository window.  
| Built-in | The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide. |
**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

| Where clauses for (for UPDATE and DELETE only) | Enter a clause to filter the data to be updated or deleted during the update or delete operations. |
| Default Table Name | Enter the default table name, between double quotation marks. |
| Default Schema Name | Enter the default schema name, between double quotation marks. |
| Table name from connection name is variable | Select this check box when the name of the connection to this component is set to a variable, such as a context variable. |
| Use different table name | Select this check box to define a different output table name, between double quotation marks, in the **Table name** field which appears. |
| Mapping | Specify the metadata mapping file for the database to be used. The metadata mapping file is used for the data type conversion between database and Java. For more information about the metadata mapping, see the related documentation for Type mapping. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |

---

691
## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th><strong>tELTGreenplumOutput</strong> is to be used along with the <strong>tELTGreenplumMap</strong>. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong> Note that the ELT components do not handle actual data flow but only schema information.</td>
</tr>
</tbody>
</table>

## Related scenarios

- **Scenario: Mapping data using a simple implicit join** on page 682
- **Scenario 1: Aggregating table columns and filtering** on page 744
- **Scenario 2: ELT using an Alias table** on page 748
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTHiveInput

Replicates the schema, which the tELTHiveMap component that follows will use, of the input Hive table.

The three ELT Hive components are closely related, in terms of their operating conditions. These components should be used to handle Hive DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

This component provides, for the tELTHiveMap component that follows, the input schema of the Hive table to be used.

**tELTHiveInput Standard properties**

These properties are used to configure tELTHiveInput running in the Standard Job framework.

The Standard tELTHiveInput component belongs to the ELT family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema</th>
<th>A schema is a row description. It defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

**Edit schema**

Click **Edit schema** to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Default table name**
Enter the name of the input table to be used.
**Default schema name**
Enter the name of the database schema to which the input table to be used is related.

**Advanced settings**

**tStat Catcher Statistics**
Select this check box to collect log data at the component level.

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press `Ctrl + Space` to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

**tELTHiveMap** is used along with a **tELTHiveInput** and **tELTHiveOutput**. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.

If the Studio used to connect to a Hive database is operated on Windows, you must manually create a folder called *tmp* in the root of the disk where this Studio is installed.

**Note:**
The ELT components do not handle actual data flow but only schema information.

**Usage with Dataproc**
The ELT Hive components require Tez to be installed on the Google Cloud Dataproc cluster to be used.

- Use the initialization action explained in this Google Cloud Platform documentation: [Apache Tez on Dataproc](#).

- For more details about the general concept of the initialization actions in a Google Cloud Dataproc cluster, see the related Google documentation: [Initialization actions](#).
Related scenarios

- Scenario: Joining table columns and writing them into Hive on page 707
- Aggregating Snowflake data using context variables as table and connection names on page 724
tELTHiveMap

Builds graphically the Hive QL statement in order to transform data.

The three ELT Hive components are closely related, in terms of their operating conditions. These components should be used to handle Hive DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

This component uses the tables provided as input, to feed the parameter in the built statement. The statement can include inner or outer joins to be implemented between tables or between one table and its aliases.

**tELTHiveMap Standard properties**

These properties are used to configure tELTHiveMap running in the Standard Job framework.

The Standard tELTHiveMap component belongs to the ELT family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

Connection configuration:

- When you use this component with Google Dataproc:

<table>
<thead>
<tr>
<th><strong>Project identifier</strong></th>
<th>Enter the ID of your Google Cloud Platform project. If you are not certain about your project ID, check it in the Manage Resources page of your Google Cloud Platform services.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cluster identifier</strong></td>
<td>Enter the ID of your Dataproc cluster to be used.</td>
</tr>
<tr>
<td><strong>Region</strong></td>
<td>Enter the geographic zones in which the computing resources are used and your data is stored and processed. If you do not need to specify a particular region, leave the default value global. For further information about the available regions and the zones each region groups, see Regions and Zones.</td>
</tr>
<tr>
<td><strong>Google Storage staging bucket</strong></td>
<td>As a Talend Job expects its dependent jar files for execution, specify the Google Storage directory to which these jar files are transferred so that your Job can access these files at execution. The directory to be entered must end with a slash (/). If not existing, the directory is created on the fly but the bucket to be used must already exist.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Fill this field with the name of the database.</td>
</tr>
<tr>
<td><strong>Provide Google Credentials in file</strong></td>
<td>Leave this check box clear, when you launch your Job from a given machine in which Google Cloud SDK has been installed and authorized to use your user account.</td>
</tr>
</tbody>
</table>
credentials to access Google Cloud Platform. In this situation, this machine is often your local machine. For further information about this Google Credentials file, see the administrator of your Google Cloud Platform or visit Google Cloud Platform Auth Guide.

- When you use this component with HDInsight:

| WebHCat configuration | Enter the address and the authentication information of the Microsoft HD Insight cluster to be used. For example, the address could be your_hdinsight_cluster_name.azurehdinsight.net and the authentication information is your Azure account name: ychen. The Studio uses this service to submit the Job to the HD Insight cluster.

In the Job result folder field, enter the location in which you want to store the execution result of a Job in the Azure Storage to be used. |

| HDInsight configuration | Enter the authentication information of the HD Insight cluster to be used. |

| Windows Azure Storage configuration | Enter the address and the authentication information of the Azure Storage account to be used. In this configuration, you do not define where to read or write your business data but define where to deploy your Job only. Therefore always use the Azure Blob Storage system for this configuration.

In the Container field, enter the name of the container to be used.

In the Deployment Blob field, enter the location in which you want to store the current Job and its dependent libraries in this Azure Storage account. |

| Database | Fill this field with the name of the database. |

- When you use the other distributions:

| Connection mode | Select a connection mode from the list. The options vary depending on the distribution you are using. |

| Hive server | Select the Hive server through which you want the Job using this component to execute queries on Hive.

This Hive server list is available only when the Hadoop distribution to be used such as HortonWorks Data Platform V1.2.0 (Bimota) supports HiveServer2. It allows you to select HiveServer2 (Hive 2), the server that better support concurrent connections of multiple clients than HiveServer (Hive 1).

For further information about HiveServer2, see https://cwiki.apache.org/confluence/display/Hive/Setting+Up+HiveServer2. |

<p>| Host | Database server IP address. |</p>
<table>
<thead>
<tr>
<th><strong>Port</strong></th>
<th>Listening port number of DB server.</th>
</tr>
</thead>
</table>
| **Database** | Fill this field with the name of the database.  
  **Note:**  
  This field is not available when you select **Embedded** from the **Connection mode** list. |
| **Username and Password** | DB user authentication data.  
  To enter the password, click the […] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| **Use kerberos authentication** | If you are accessing a Hive Metastore running with Kerberos security, select this check box and then enter the relevant parameters in the fields that appear.  
  - If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in **Connecting to a security-enabled MapR** on page 1652.  
  Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.  
  The values of the following parameters can be found in the **hive-site.xml** file of the Hive system to be used.  
  1. **Hive principal** uses the value of **hive.metastore.kerberos.principal**. This is the service principal of the Hive Metastore.  
  2. **HiveServer2 local user principal** uses the value of **hive.server2.authentication.kerberos.principal**.  
  3. **HiveServer2 local user keytab** uses the value of **hive.server2.authentication.kerberos.keytab**  
  4. **Metastore URL** uses the value of **javax.jdo.option.ConnectionURL**. This is the JDBC connection string to the Hive Metastore.  
  5. **Driver class** uses the value of **javax.jdo.option.ConnectionDriverName**. This is the name of the driver for the JDBC connection.  
  6. **Username** uses the value of **javax.jdo.option.ConnectionUserName**. This, as well as the Password parameter, is the user credential for connecting to the Hive Metastore.  
  7. **Password** uses the value of **javax.jdo.option.ConnectionPassword**.  
  For the other parameters that are displayed, please consult the Hadoop configuration files they belong to.
For example, the **Namenode principal** can be found in the `hdfs-site.xml` file or the `hdfs-default.xml` file of the distribution you are using.

This check box is available depending on the Hadoop distribution you are connecting to.

### Use a keytab to authenticate

Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used.

### Use SSL encryption

Select this check box to enable the SSL or TLS encrypted connection.

Then in the fields that are displayed, provide the authentication information:

- In the **Trust store path** field, enter the path, or browse to the TrustStore file to be used. By default, the supported TrustStore types are **JKS** and **PKCS 12**.

- To enter the password, click the `[...]` button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

This feature is available only to the HiveServer2 in the **Standalone** mode of the following distributions:

- Hortonworks Data Platform 2.0 +
- Cloudera CDH4 +
- Pivotal HD 2.0 +
- Amazon EMR 4.0.0 +

### Set Resource Manager

Select this check box and in the displayed field, enter the location of the ResourceManager of your distribution. For example, `tal-qa114.talend.lan:8050`.

Then you can continue to set the following parameters depending on the configuration of the Hadoop cluster to be used (if you leave the check box of a parameter clear, then at runtime, the configuration about this parameter in the Hadoop cluster to be used will be ignored).
1. Select the **Set resourcemanager scheduler address** check box and enter the Scheduler address in the field that appears.

2. Select the **Set jobhistory address** check box and enter the location of the JobHistory server of the Hadoop cluster to be used. This allows the metrics information of the current Job to be stored in that JobHistory server.

3. Select the **Set staging directory** check box and enter this directory defined in your Hadoop cluster for temporary files created by running programs. Typically, this directory can be found under the `yarn.app.mapreduce.am.staging-dir` property in the configuration files such as `yarn-site.xml` or `mapred-site.xml` of your distribution.

4. Allocate proper memory volumes to the **Map** and the **Reduce** computations and the **ApplicationMaster** of YARN by selecting the **Set memory** check box in the **Advanced settings** view.

5. Select the **Set Hadoop user** check box and enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed.

6. Select the **Use datanode hostname** check box to allow the Job to access datanodes via their hostnames. This actually sets the `dfs.client.use.datanode.hostname` property to `true`. When connecting to a S3N filesystem, you must select this check box.

For further information about these parameters, see the documentation or contact the administrator of the Hadoop cluster to be used.


**Set NameNode URI**

Select this check box and in the displayed field, enter the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, assuming that you have chosen a machine called `masternode` as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS.

The other properties:

| Property type            | Either **Built-In** or **Repository**.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In</strong>:</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong>:</td>
<td>Select the repository file where the</td>
</tr>
<tr>
<td></td>
<td>properties are stored.</td>
</tr>
</tbody>
</table>

| Use an existing connection | Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined. |

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

| ELT Hive Map editor | The ELT Map editor helps you to define the output schema as well as build graphically the Hive QL statement to be executed. The column names of schema can be different from the column names in the database. If you use context variables in the **Expression** column in the Map editor to map the input and the output schemas, put single quotation marks around these context variables, for example, `context.v_erp Name`. |

| Style link | Select the way in which links are displayed.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Auto</strong>:</td>
<td>By default, the links between the input and output schemas and the Web service parameters are in the form of curves.</td>
</tr>
<tr>
<td><strong>Bezier curve</strong>:</td>
<td>Links between the schema and the Web service parameters are in the form of curve.</td>
</tr>
<tr>
<td><strong>Line</strong>:</td>
<td>Links between the schema and the Web service parameters are in the form of straight lines. This option slightly optimizes performance.</td>
</tr>
</tbody>
</table>

| Distribution | Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration: |
• If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

• If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

• The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the Import from existing version option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

Note:
In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.
For a step-by-step example about how to connect to a custom distribution and share this connection, see Connecting to a custom Hadoop distribution on page 1589.

### Hive version
Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.

### Execution engine
Select this check box and from the drop-down list, select the framework you need to use to run the Job.

This list is available only when you are using the **Embedded** mode for the Hive connection and the distribution you are working with is among the following ones:

- Hortonworks: V2.1 and V2.2.
- MapR: V4.0.1.
- Custom: this option allows you connect to a distribution supporting Tez but not officially supported by Talend.

Before using Tez, ensure that the Hadoop cluster you are using supports Tez. You will need to configure the access to the relevant Tez libraries via the **Advanced settings** view of this component.

For further information about Hive on Tez, see Apache’s related documentation in https://cwiki.apache.org/confluence/display/Hive/Hive+on+Tez. Some examples are presented there to show how Tez can be used to gain performance over MapReduce.

When you need to enable Hive components to access HBase:

These parameters are available only when the **Use an existing connection** check box is clear.

#### Store by HBase
Select this check box to display the parameters to be set to allow the Hive components to access HBase tables:

- Once this access is configured, you will be able to use, in `tHiveRow` and `tHiveInput`, the Hive QL statements to read and write data in HBase.

- If you are using the Kerberos authentication, you need to define the HBase related principals in the corresponding fields that are displayed.

For further information about this access involving Hive and HBase, see Apache’s Hive documentation about Hive/HBase integration.

#### Zookeeper quorum
Type in the name or the URL of the Zookeeper service you use to coordinate the transaction between your Studio and your database. Note that when you configure the Zookeeper, you might need to explicitly set the `zookeeper.znode.parent` property to define the path to the root znode that contains all the znodes created and used...
by your database; then select the Set Zookeeper znode parent check box to define this property.

**Zookeeper client port**
Type in the number of the client listening port of the Zookeeper service you are using.

**Define the jars to register for HBase**
Select this check box to display the Register jar for HBase table, in which you can register any missing jar file required by HBase, for example, the Hive Storage Handler, by default, registered along with your Hive installation.

**Register jar for HBase**
Click the [+] button to add rows to this table, then, in the Jar name column, select the jar file(s) to be registered and in the Jar path column, enter the path(s) pointing to that or those jar file(s).

### Advanced settings

**Tez lib**
Select how the Tez libraries are accessed:

- **Auto install**: at runtime, the Job uploads and deploys the Tez libraries provided by the Studio into the directory you specified in the Install folder in HDFS field, for example, /tmp/usr/tez.
  
  If you have set the tez.lib.uris property in the properties table, this directory overrides the value of that property at runtime. But the other properties set in the properties table are still effective.

- **Use exist**: the Job accesses the Tez libraries already deployed in the Hadoop cluster to be used. You need to enter the path pointing to those libraries in the Lib path (folder or file) field.

- **Lib jar**: this table appears when you have selected Auto install from the Tez lib list and the distribution you are using is Custom. In this table, you need to add the Tez libraries to be uploaded.

**Temporary path**
If you do not want to set the Jobtracker and the NameNode when you execute the query `select * from your_table_name`, you need to set this temporary path. For example, /C:/select_all in Windows.

**Hadoop properties**
Talend Studio uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- **Note**: if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.
For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache's Hadoop documentation on http://hadoop.apache.org/docs and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml.
- Apache also provides a page to list the Hive-related properties: https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties.

### Hive properties

Talend Studio uses a default configuration for its engine to perform operations in a Hive database. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones. For further information for Hive dedicated properties, see https://cwiki.apache.org/confluence/display/Hive/AdminManual+Configuration.

- If you need to use Tez to run your Hive Job, add `hive.execution.engine` to the **Properties** column and `Tez` to the **Value** column, enclosing both of these strings in double quotation marks.
- Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.

### Mapred job map memory mb and Mapred job reduce memory mb

You can tune the map and reduce computations by selecting the **Set memory** check box to set proper memory allocations for the computations to be performed by the Hadoop system.

In that situation, you need to enter the values you need in the **Mapred job map memory mb** and the **Mapred job reduce memory mb** fields, respectively. By default, the values are both 1000 which are normally appropriate for running the computations.

### Path separator in server

Leave the default value of the **Path separator in server** as it is, unless you have changed the separator used by your Hadoop distribution's host machine for its PATH variable or in other words, that separator is not a colon (:). In that situation, you must change this value to the one you are using in that host.

### tStatCatcher Statistics

Select this check box to collect log data at the component level.

### Temporary path

If you do not want to set the Jobtracker and the NameNode when you execute the query `select *`.
**Global Variables**

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

**Usage rule**

*tELTHiveMap* is used along with a *tELTHiveInput* and *tELTHiveOutput*. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.

If the Studio used to connect to a Hive database is operated on Windows, you must manually create a folder called *tmp* in the root of the disk where this Studio is installed.

**Note:**
The ELT components do not handle actual data flow but only schema information.

**Usage with Dataproc**

The ELT Hive components require Tez to be installed on the Google Cloud Dataproc cluster to be used.

- Use the initialization action explained in this Google Cloud Platform documentation: *Apache Tez on Dataproc*.
- For more details about the general concept of the initialization actions in a Google Cloud Dataproc cluster, see the related Google documentation: *Initialization actions*.

**Dynamic settings**

Click the [*] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.
The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: *no MapRClient in java.library.path.*

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the [Preferences] dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the **Data viewer** to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

---

**Scenario: Joining table columns and writing them into Hive**

This scenario applies only to a Talend solution with Big Data.

This scenario uses a four-component Job to join the columns selected from two Hive tables and write them into another Hive table.
Preparing the Hive tables

Procedure

1. Create the Hive table you want to write data in. In this scenario, this table is named as `agg_result`, and you can create it using the following statement in `tHiveRow`:
   ```sql
   create table agg_result
   (id int, name string, address string, sum1 string, postal string, state string, capital string, mostpopulouscity string) partitioned by (type string) row format delimited fields terminated by ';' location '/user/ychen/hive/table/agg_result'
   ```
   In this statement, `/user/ychen/hive/table/agg_result` is the directory used in this scenario to store this created table in HDFS. You need to replace it with the directory you want to use in your environment.

   For further information about `tHiveRow`, see `tHiveRow` on page 1639.

2. Create two input Hive tables containing the columns you want to join and aggregate these columns into the output Hive table, `agg_result`. The statements to be used are:
   ```sql
   create table customer (id int, name string, address string, idState int, id2 int, regTime string, registerTime string, sum1 string, sum2 string) row format delimited fields terminated by ';' location '/user/ychen/hive/table/customer'
   ```
   ```sql
   create table state_city (id int, postal string, state string, capital int, mostpopulouscity string) row format delimited fields terminated by ';' location '/user/ychen/hive/table/state_city'
   ```

3. Use `tHiveRow` to load data into the two input tables, `customer` and `state_city`. The statements to be used are:
   ```sql
   "LOAD DATA LOCAL INPATH 'C:/tmp/customer.csv' OVERWRITE INTO TABLE customer" and "LOAD DATA LOCAL INPATH 'C:/tmp/State_City.csv' OVERWRITE INTO TABLE state_city"
   ```
   The two files, `customer.csv` and `State_City.csv`, are two local files we created for this scenario. You need to create your own files to provide data to the input Hive tables. The data schema of each file should be identical with their corresponding table.

   You can use `tRowGenerator` and `tFileOutputDelimited` to create these two files easily. For further information about these two components, see `tRowGenerator` on page 3281 and `tFileOutputDelimited` on page 1110.

   For further information about the Hive query language, see https://cwiki.apache.org/confluence/display/Hive/LanguageManual.
**Linking the components**

**Procedure**

1. In the **Integration** perspective of **Talend Studio**, create an empty Job from the **Job Designs** node in the **Repository** tree view.  
   For further information about how to create a Job, see **Talend Studio User Guide**.
2. Drop two **tELTHiveInput** components and **tELTHiveMap** and **tELTHiveOutput** onto the workspace.
3. Connect them using the **Row > Main** link.
   Each time when you connect two components, a wizard pops up to prompt you to name the link you are creating. This name must be the same as that of the Hive table you want the active component to process. In this scenario, the input tables the two **tELTHiveInput** components will handle are **customer** and **state_city** and the output table **tELTHiveOutput** will handle is **agg_result**.

**Configuring the input schemas**

**Procedure**

1. Double-click the **tELTHiveInput** component using the **customer** link to open its **Component** view.
   ![Component view of tELTHiveInput](image)

2. Click the [...] button next to **Edit schema** to open the schema editor.
3. Click the + button as many times as required to add columns and rename them to replicate the schema of the **customer** table we created earlier in Hive.
4. In the **Default table name** field, enter the name of the input table, *customer*, to be processed by this component.

5. Double-click the other tELTHiveInput component using the *state_city* link to open its **Component** view.

6. Click the [...] button next to **Edit schema** to open the schema editor.

7. Click the button as many times as required to add columns and rename them to replicate the schema of the *state_city* table we created earlier in Hive.
8. In the **Default table name** field, enter the name of the input table, `state_city`, to be processed by this component.

**Mapping the input and the output schemas**

**Configuring the connection to Hive**

**Procedure**

1. Click **tELTHiveMap**, then, click **Component** to open its **Component** view.

2. In the **Version** area, select the Hadoop distribution you are using and the Hive version.

3. In the **Connection mode** list, select the connection mode you want to use. If your distribution is HortonWorks, this mode is **Embedded** only.
4. In the Host field and the Port field, enter the authentication information for the component to connect to Hive. For example, the host is talend-hdp-all and the port is 9083.

5. Select the Set Jobtracker URI check box and enter the location of the Jobtracker. For example, talend-hdp-all:50300.

6. Select the Set NameNode URI check box and enter the location of the NameNode. For example, hdfs://talend-hdp-all:8020. If you are using WebHDFS, the location should be webhdfs://master:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS.

Mapping the schemas

Procedure

1. Click ELT Hive Map Editor to map the schemas.

![ELT Hive Map Editor](image)

2. On the input side (left in the figure), click the Add alias button to add the table to be used.

![Add alias](image)

3. In the pop-up window, select the customer table, then click OK.

4. Repeat the operations to select the state_city table.

5. Drag and drop the idstate column from the customer table onto the id column of the state_city table. Thus an inner join is created automatically.

6. On the output side (the right side in the figure), the agg_result table is empty at first. Click at the bottom of this side to add as many columns as required and rename them to replicate the schema of the agg_result table you created earlier in Hive.
The type column is the partition column of the agg_result table and should not be replicated in this schema. For further information about the partition column of the Hive table, see the Hive manual.

7. From the customer table, drop id, name, address, and sum1 to the corresponding columns in the agg_result table.

8. From the state_city table, drop postal, state, capital and mostpopulouscity to the corresponding columns in the agg_result table.

In this scenario, context variables are not used in the Expression column in the Map editor. If you use context variables, put them in single quotation marks. For example:

9. Click OK to validate these changes.

**Configuring the output schema**

**Procedure**

1. Double-click tELTHiveOutput to open its Component view.
2. If this component does not have the same schema of the preceding component, a warning icon appears. In this case, click the **Sync columns** button to retrieve the schema from the preceding one and once done, the warning icon disappears.

3. In the **Default table name** field, enter the output table you want to write data in. In this example, it is `agg_result`.

4. In the **Field partition** table, click ![add row icon] to add one row. This allows you to write data in the partition column of the `agg_result` table.
   
   This partition column was defined the moment we created the `agg_result` table using `partitioned by (type string)` in the Create statement presented earlier. This partition column is `type`, which describes the type of a customer.

5. In **Partition column**, enter `type` without any quotation marks and in **Partition value**, enter `prospective` in single quotation marks.

**Executing the Job**

**Procedure**

Press F6 to run this Job.

**Results**

Once done, verify `agg_result` in Hive using, for example,

```
select * from agg_result;
```
This figure presents only a part of the table. You can find that the selected input columns are aggregated and written into the `agg_result` table and the partition column is filled with the value `prospective`.

### Related scenarios

- **Scenario**: Joining table columns and writing them into Hive on page 707
- **Scenario**: Mapping data using a subquery on page 800, a related scenario using subquery
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
**tELTHiveOutput**

Works alongside tELTHiveMap to write data into the Hive table.

The three ELT Hive components are closely related, in terms of their operating conditions. These components should be used to handle Hive DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

This component executes the query built by the preceding tELTHiveMap component to write data into the specified Hive table.

**tELTHiveOutput Standard properties**

These properties are used to configure tELTHiveOutput running in the Standard Job framework.

The Standard tELTHiveOutput component belongs to the ELT family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Action on data</th>
<th>Select the action to be performed on the data to be written in the Hive table. With the Insert option, the data to be written in the Hive table will be appended to the existing data if there is any.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>A schema is a row description. It defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-In:</strong> You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Edit schema</td>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No.</td>
</tr>
</tbody>
</table>
upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Default table name</th>
<th>Enter the default name of the output table you want to write data in.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default schema name</td>
<td>Enter the name of the default database schema to which the output table to be used is related to.</td>
</tr>
<tr>
<td>Table name from connection name is variable</td>
<td>Select this check box when the name of the connection to this component is set to a variable, such as a context variable.</td>
</tr>
</tbody>
</table>
| Use different table name | Select this check box to define a different output table name, between double quotation marks, in the **Table name** field that appears.  
If this table is related to a different database schema from the default one, you also need to enter the name of that database schema. The syntax is `schema_name.table_name`. |
| The target table uses the Parquet format | If the table in which you need to write data is a Parquet table, select this check box.  
Then from the **Compression** list that appears, select the compression mode you need to use to handle the Parquet file. The default mode is **Uncompressed**. |
| Field Partition | In **Partition Column**, enter the name, without any quotation marks, of the partition column of the Hive table you want to write data in.  
In **Partition Value**, enter the value you want to use, in single quotation marks, for its corresponding partition column. |
| Mapping | Specify the metadata mapping file for the database to be used. The metadata mapping file is used for the data type conversion between database and Java. For more information about the metadata mapping, see the related documentation for Type mapping. |

### Advanced settings

**tStatCatcher Statistics**  
Select this check box to collect log data at the component level.

### Global Variables

**Global Variables**  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

#### Usage rule

**tELTHiveMap** is used along with a **tELTHiveInput** and **tELTHiveOutput**. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.

If the Studio used to connect to a Hive database is operated on Windows, you must manually create a folder called **tmp** in the root of the disk where this Studio is installed.

**Note:**

The ELT components do not handle actual data flow but only schema information.

#### Usage with Dataproc

The ELT Hive components require Tez to be installed on the Google Cloud Dataproc cluster to be used.

- Use the initialization action explained in this Google Cloud Platform documentation: [Apache Tez on Dataproc](https://cloud.google.com/dataproc/docs/quickstart-initialization-action).
- For more details about the general concept of the initialization actions in a Google Cloud Dataproc cluster, see the related Google documentation: [Initialization actions](https://cloud.google.com/dataproc/docs/quickstart-initialization-action).

### Related scenarios

- **Scenario: Joining table columns and writing them into Hive** on page 707
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
**tELTJDBCInput**

Adds as many Input tables as required for the SQL statement to be executed.

The three ELT JDBC components are closely related, in terms of their operating conditions. These components should be used to handle JDBC DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

Note that it is highly recommended to use the ELT components for a specific type of database (if any) instead of the ELT JDBC components. For example, for Teradata, it is recommended to use the tELTTeradataInput, tELTTeradataMap and tELTTeradataOutput components instead.

**tELTJDBCInput Standard properties**

These properties are used to configure tELTJDBCInput running in the Standard Job framework.

The Standard tELTJDBCInput component belongs to the ELT family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description, it defines the nature and number of fields to be processed. The schema is either built-in or remotely stored in the Repository. The Schema defined is then passed on to the ELT Mapper to be included to the Insert SQL statement. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td>Built-in: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td>Repository: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Default Table Name</td>
<td>Type in the default table name.</td>
</tr>
<tr>
<td>Default Schema Name</td>
<td>Type in the default schema name.</td>
</tr>
</tbody>
</table>
Mapping

Specify the metadata mapping file for the database to be used. The metadata mapping file is used for the data type conversion between database and Java. For more information about the metadata mapping, see the related documentation for Type mapping.

Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

**Usage rule**

tELTJDBCInput is to be used along with the tELTJDBCMAP. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.

**Note:**

Note that the ELT components do not handle actual data flow but only schema information.

Related scenarios

- **Scenario 1: Aggregating table columns and filtering** on page 744
- **Scenario 2: ELT using an Alias table** on page 748
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTJDBCMap

Uses the tables provided as input to feed the parameter in the built SQL statement. The statement can include inner or outer joins to be implemented between tables or between one table and its aliases.

The three ELT JDBC components are closely related, in terms of their operating conditions. These components should be used to handle JDBC DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

Note that it is highly recommended to use the ELT components for a specific type of database (if any) instead of the ELT JDBC components. For example, for Teradata, it is recommended to use the tELTTeradataInput, tELTTeradataMap and tELTTeradataOutput components instead.

**tELTJDBCMap Standard properties**

These properties are used to configure tELTJDBCMap running in the Standard Job framework.

The Standard tELTJDBCMap component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td><strong>1.</strong> In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td><strong>2.</strong> In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td></td>
<td>For an example about how to share a database connection across Job levels, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ELT JDBC Map Editor</th>
<th>The ELT Map editor allows you to define the output schema and make a graphical build of the SQL statement to be executed. The column names of schema can be different from the column names in the database.</th>
</tr>
</thead>
</table>

| Style link | Select the way in which links are displayed. |
Auto: By default, the links between the input and output schemas and the Web service parameters are in the form of curves.

Bezier curve: Links between the schema and the Web service parameters are in the form of curve.

Line: Links between the schema and the Web service parameters are in the form of straight lines.

This option slightly optimizes performance.

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>• Repository: Select the repository file where the properties are stored.</td>
</tr>
</tbody>
</table>

| JDBC URL      | The JDBC URL of the database to be used. For example, the JDBC URL for the Amazon Redshift database is jdbc:redshift://endpoint:port/database. |

| Driver JAR    | Complete this table to load the driver JARs needed. To do this, click the [+ ] button under the table to add as many rows as needed, each row for a driver JAR, then select the cell and click the [...] button at the right side of the cell to open the Select Module wizard from which you can select the driver JAR of your interest. For example, the driver jar RedshiftJDBC41-1.1.13.1013.jar for the Redshift database. |

| Class name    | Enter the class name for the specified driver between double quotation marks. For example, for the RedshiftJDBC41-1.1.13.1013.jar driver, the name to be entered is com.amazon.redshift.jdbc41.Driver. |

| Username and Password | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |

| Mapping        | Specify the metadata mapping file for the database to be used. The metadata mapping file is used for the data type conversion between database and Java. For more information about the metadata mapping, see the related documentation for Type mapping. |

**Advanced settings**

| Additional JDBC parameters | Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the Use an existing connection check box in the Basic settings. |
tStatCatcher Statistics

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

Global Variables

Global Variables

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule
tELTJDBCMap is used along with tELTJDBCInput and tELTJDBCOutput. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.

Note:

Note that the ELT components do not handle actual data flow but only schema information.

Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.
Aggregating Snowflake data using context variables as table and connection names

Here is an example of aggregating Snowflake data from two source tables STUDENT and TEACHER to one target table FULLINFO using the ELT JDBC components. In this example, all input and output table names and connection names are set to context variables.

Creating a Job for aggregating Snowflake data

Before you begin

- A new Job has been created and the context variables SourceTableS with the value STUDENT, SourceTableT with the value TEACHER, and TargetTable with the value FULLINFO have been added to the Job. For more information about how to use context variables, see the related documentation about using contexts and variables.

- The source table STUDENT with three columns, SID and TID of NUMBER(38,0) type and SNAME of VARCHAR(50) type, has been created in Snowflake, and the following data has been written into the table:

<table>
<thead>
<tr>
<th>#SID;SNAME;TID</th>
</tr>
</thead>
<tbody>
<tr>
<td>11;Alex;22</td>
</tr>
<tr>
<td>12;Mark;23</td>
</tr>
<tr>
<td>13;Stephane;21</td>
</tr>
<tr>
<td>14;Cedric;22</td>
</tr>
<tr>
<td>15;Bill;21</td>
</tr>
<tr>
<td>16;Jack;23</td>
</tr>
<tr>
<td>17;John;22</td>
</tr>
<tr>
<td>18;Andrew;23</td>
</tr>
</tbody>
</table>
• The source table **TEACHER** with three columns, **TID** of **NUMBER(38,0)** type and **TNAME** and **TPHONE** of **VARCHAR(50)** type, has been created in Snowflake, and the following data has been written into the table.

```
#TID;TNAME;TPHONE
21;Peter;+86 15812343456
22;Michael;+86 13178964532
23;Candice;+86 13923187456
```

**Procedure**

1. Add a **tSnowflakeConnection** component, a **tSnowflakeClose** component, two **tELTJDBCInput** components, a **tELTJDBCMap** component, and a **tELTJDBCOutput** component to your Job.

2. On the **Basic setting** view of the first **tELTJDBCInput** component, enter the name of the first source table in the **Default Table Name** field. In this example, it is the context variable `context.SourceTableS`.

3. Do the same to set the value of the default table name for the second **tELTJDBCInput** component and the **tELTJDBCOutput** component to `context.SourceTableT` and `context.TargetTable` respectively.

4. Link the first **tELTJDBCInput** component to the **tELTJDBCMap** component using the **Link > context.SourceTableS (Table)** connection.

5. Link the second **tELTJDBCInput** component to the **tELTJDBCMap** component using the **Link > context.SourceTableT (Table)** connection.

6. Link the **tELTJDBCMap** component to the **tELTJDBCOutput** component using the **Link > *New Output* (Table)** connection. The link will be renamed automatically to `context.TargetTable (Table)`.

7. Link the **tSnowflakeConnection** component to the **tELTJDBCMap** component using a **Trigger > On Subjob Ok** connection.

8. Do the same to link the **tELTJDBCMap** component to the **tSnowflakeClose** component.

**Connecting to Snowflake**

Configure the **tSnowflakeConnection** component to connect to Snowflake.

**Procedure**

1. Double-click the **tSnowflakeConnection** component to open its **Basic settings** view.

2. In the **Account** field, enter the account name assigned by Snowflake.

3. In the **User Id** and the **Password** fields, enter the authentication information accordingly.

   Note that this user ID is your user login name. If you do not know your user login name yet, ask the administrator of your Snowflake system for details.
4. In the **Warehouse** field, enter the name of the data warehouse to be used in Snowflake.
5. In the **Schema** field, enter the name of the database schema to be used.
6. In the **Database** field, enter the name of the database to be used.

**Configuring the input components for aggregating Snowflake data**

**Procedure**
1. Double-click the first tELTJDBCInput component to open its **Basic settings** view.
2. Click the [...] button next to **Edit schema** and in the schema dialog box displayed, define the schema by adding three columns, **SID** and **TID** of INT type and **SNAME** of VARCHAR type.
3. Select **Mapping Snowflake** from the **Mapping** drop-down list.
4. Repeat the previous steps to configure the second tELTJDBCInput component, and define its schema by adding three columns, **TID** of INT type and **TNAME** and **TPHONE** of VARCHAR type.

**Configuring the output component for aggregating Snowflake data**

**Procedure**
1. Double-click the tELTJDBCOutput component to open its **Basic settings** view.
2. Select **Create table** from the **Action on table** drop-down list to create the target table.
3. Select the **Table name from connection name is variable** check box.
4. Select **Mapping Snowflake** from the **Mapping** drop-down list.

**Configuring the map component for aggregating Snowflake data**

**Procedure**
1. Click the tELTJDBCMap component to open its **Basic settings** view.
2. Select the **Use an existing connection** check box and from the **Component List** displayed, select the connection component you have configured to open the Snowflake connection.
3. Select **Mapping Snowflake** from the **Mapping** drop-down list.
4. Click the [...] button next to **ELT JDBC Map Editor** to open its map editor.
5. Add the first input table `context.SourceTableS` by clicking the [+] button in the upper left corner of the map editor and then selecting the relevant table name from the drop-down list in the pop-up dialog box.
6. Do the same to add the second input table `context.SourceTableT`.
7. Drag the column TID from the first input table context.SourceTableS and drop it onto the corresponding column TID in the second input table context.SourceTableT.

8. Drag all columns from the input table context.SourceTableS and drop them onto the output table context.TargetTable in the upper right panel.

9. Do the same to drag two columns TNAME and TPHONE from the input table context.SourceTableT and drop them onto the bottom of the output table. When done, click OK to close the map editor.

10. Click the Sync columns button on the Basic settings view of the tELTJDBCOutput component to set its schema.

**Closing the Snowflake connection**

Configure the tSnowflakeClose component to close the connection to Snowflake.

**Procedure**

1. Double-click the tSnowflakeClose component to open the Component tab.

2. From the Connection Component drop-down list, select the component that opens the connection you need to close, tSnowflakeConnection_1 in this example.

**Executing the Job for aggregating Snowflake data**

**Procedure**

1. Press Ctrl + S to save the Job.

2. Press F6 to execute the Job.

```
[statistics] connecting to socket on port 3572
[statistics] connected
Inserting with:
INSERT INTO FULLINFO(SID,SNAME,TID,TNAME,TPHONE) (SELECT STUDENT.SID, STUDENT.SNAME, STUDENT.TID, TEACHER.TNAME, TEACHER.TPHONE FROM STUDENT, TEACHER WHERE TEACHER.TID = STUDENT.TID )
---> 8 rows inserted.
[statistics] disconnected
```

As shown above, the Job is executed successfully and eight rows are inserted into the target table.

You can then create and run another Job to retrieve data from the target table by using the tSnowflakeInput component and the tLogRow component. You will find that the aggregated data will be displayed on the console as shown in below screenshot.

```
<table>
<thead>
<tr>
<th>SID</th>
<th>SNAME</th>
<th>TID</th>
<th>TNAME</th>
<th>TPHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Alex</td>
<td>22</td>
<td>Michael</td>
<td>+86 13178964532</td>
</tr>
<tr>
<td>12</td>
<td>Merk</td>
<td>23</td>
<td>Candice</td>
<td>+86 13923107456</td>
</tr>
<tr>
<td>13</td>
<td>Stephane</td>
<td>21</td>
<td>Peter</td>
<td>+86 13812343456</td>
</tr>
<tr>
<td>14</td>
<td>Cedric</td>
<td>22</td>
<td>Michael</td>
<td>+86 13178964532</td>
</tr>
<tr>
<td>15</td>
<td>Bill</td>
<td>21</td>
<td>Peter</td>
<td>+86 13912343456</td>
</tr>
<tr>
<td>16</td>
<td>Jack</td>
<td>23</td>
<td>Candice</td>
<td>+86 13923107456</td>
</tr>
<tr>
<td>17</td>
<td>John</td>
<td>22</td>
<td>Michael</td>
<td>+86 13178964532</td>
</tr>
<tr>
<td>18</td>
<td>Andrew</td>
<td>23</td>
<td>Candice</td>
<td>+86 13923107456</td>
</tr>
</tbody>
</table>
```

For more information about how to retrieve data from Snowflake, see the example Writing data into and reading data from a Snowflake table on page 3548.
Related scenarios

- **Scenario 1**: Aggregating table columns and filtering on page 744.
- **Scenario 2**: ELT using an Alias table on page 748.
- **Scenario**: Mapping data using a subquery on page 800, a related scenario using subquery
tELTJDBCOutput

Carries out the action on the table specified and inserts the data according to the output schema defined in the ELT Mapper.

The three ELT JDBC components are closely related, in terms of their operating conditions. These components should be used to handle JDBC DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

Note that it is highly recommended to use the ELT components for a specific type of database (if any) instead of the ELT JDBC components. For example, for Teradata, it is recommended to use the tELTTeradataInput, tELTTeradataMap and tELTTeradataOutput components instead.

### tELTJDBCOutput Standard properties

These properties are used to configure tELTJDBCOutput running in the Standard Job framework.

The Standard tELTJDBCOutput component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Action on table</th>
<th>Select an operation to be performed on the table defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Drop and create table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Create table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Create table if does not exist</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Drop table if exist and create</strong>: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Clear table</strong>: The table content is deleted. You have the possibility to rollback the operation.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Truncate table</strong>: The table content is deleted. You do not have the possibility to rollback the operation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action on data</th>
<th>On the data of the table defined, you can perform the following operation:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Insert</strong>: Adds new entries to the table. If duplicates are found, Job stops.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update</strong>: Updates entries in the table.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Delete</strong>: Deletes the entries which correspond to the entry flow.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next</th>
</tr>
</thead>
</table>
component. The schema is either built-in or remotely stored in the Repository.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

| **Build-in**: The schema is created and stored locally for this component only. Related topic: see **Talend Studio User Guide**. |
|**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see **Talend Studio User Guide**. |

| **Where clauses for** (for UPDATE and DELETE only) | Enter a clause to filter the data to be updated or deleted during the update or delete operations. |
| **Default Table Name** | Enter the default table name, between double quotation marks. |
| **Default Schema Name** | Enter the default schema name, between double quotation marks. |
| **Table name from connection name is variable** | Select this check box when the name of the connection to this component is set to a variable, such as a context variable. |
| **Use different table name** | Select this check box to define a different output table name, between double quotation marks, in the **Table name** field which appears. |
| **Mapping** | Specify the metadata mapping file for the database to be used. The metadata mapping file is used for the data type conversion between database and Java. For more information about the metadata mapping, see the related documentation for **Type mapping**. |

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |
Global Variables

Global Variables

**NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

**NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

Usage

**Usage rule**

**tELTJDBCOutput** is to be used along with the **tELTJDBCMap**. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.

Note that the ELT components do not handle actual data flow but only schema information.

**Limitation**

Avoid using any keyword for the database as the table/column name or using any special character in the table/column name. If you want to, you can enclose the table/column name in a pair of " to see whether it works. For example, when you want to use the keyword *number* as an Oracle database column name, you can have the **Db Column** value in the schema editor set to "number". But note that this solution does not always work.

Related scenarios

- **Scenario 1: Aggregating table columns and filtering** on page 744
- **Scenario 2: ELT using an Alias table** on page 748
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTMSSqlInput

Adds as many Input tables as required for the most complicated Insert statement.

The three ELT MSSql components are closely related, in terms of their operating conditions. These components should be used to handle MSSql DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

Provides the table schema to be used for the SQL statement to execute.

tELTMSSqlInput Standard properties

These properties are used to configure tELTMSSqlInput running in the Standard Job framework.

The Standard tELTMSSqlInput component belongs to the ELT family.

The component in this framework is generally available.

Basic settings

| Schema and Edit schema | A schema is a row description, it defines the nature and number of fields to be processed. The schema is either built-in or remotely stored in the Repository. The Schema defined is then passed on to the ELT Mapper to be included to the Insert SQL statement. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:
| | • View schema: choose this option to view the schema only.
| | • Change to built-in property: choose this option to change the schema to Built-in for local changes.
| | • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.
| Built-in | The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.
| Repository | The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.
| Default Table Name | Type in the default table name.
| Default Schema Name | Type in the default schema name. |
Advanced settings

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**

**tELTMySSqlInput** is to be used along with the **tELTMSSsqlMap**. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.

**Note:**
Note that the ELT components do not handle actual data flow but only schema information.

Related scenarios

- **Scenario 1: Aggregating table columns and filtering** on page 744
- **Scenario 2: ELT using an Alias table** on page 748
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTMSSqlMap

Uses the tables provided as input to feed the parameter in the built statement. The statement can include inner or outer joins to be implemented between tables or between one table and its aliases.

The three ELT MSSql components are closely related, in terms of their operating conditions. These components should be used to handle MSSql DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

Helps you to build the SQL statement graphically, using the table provided as input.

**tELTMSSqlMap Standard properties**

These properties are used to configure tELTMSSqlMap running in the Standard Job framework.

The Standard tELTMSSqlMap component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
</tbody>
</table>

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>ELT MSSql Map Editor</th>
<th>The ELT Map editor allows you to define the output schema and make a graphical build of the SQL statement to be executed. The column names of schema can be different from the column names in the database.</th>
</tr>
</thead>
</table>

| Style link | Select the way in which links are displayed. **Auto:** By default, the links between the input and output schemas and the Web service parameters are in the form of curves. **Bezier curve:** Links between the schema and the Web service parameters are in the form of curve. |
**Line**: Links between the schema and the Web service parameters are in the form of straight lines. This option slightly optimizes performance.

**Property type**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the Repository file where Properties are stored. The following fields are pre-filled in using fetched data.</td>
</tr>
</tbody>
</table>

**Host**

Database server IP address.

**Port**

Listening port number of DB server.

**Database**

Name of the database.

**Username and Password**

DB user authentication data.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**Advanced settings**

**Additional JDBC parameters**

Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the Basic settings.

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

**Global Variables**

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*. 
## Usage

| Usage rule | **tELTMSqlMap** is used along with a **tELTMSqlInput** and **tELTMSqlOutput**. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.  

**i** Note:  
Note that the ELT components do not handle actual data flow but only schema information. |
|------------|--------------------------------------------------------------------------------------------------|

| Dynamic settings | Click the [*] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.  

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489.  

For more information on **Dynamic settings** and context variables, see **Talend Studio User Guide**. |

---

## Related scenarios

- **Scenario 1: Aggregating table columns and filtering** on page 744
- **Scenario 2: ELT using an Alias table** on page 748
- **Scenario: Mapping data using a subquery** on page 800, a related scenario using subquery
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTMSSqlOutput

Executes the SQL Insert, Update and Delete statement to the MSSql database

The three ELT MSSql components are closely related, in terms of their operating conditions. These components should be used to handle MSSql DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

Carries out the action on the table specified and inserts the data according to the output schema defined the ELT Mapper.

**tELTMSSqlOutput Standard properties**

These properties are used to configure tELTMSSqlOutput running in the Standard Job framework.

The Standard tELTMSSqlOutput component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Action on data</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the data of the table defined, you can perform the following operation:</td>
</tr>
<tr>
<td><strong>Insert</strong>: Adds new entries to the table. If duplicates are found, Job stops.</td>
</tr>
<tr>
<td><strong>Update</strong>: Updates entries in the table.</td>
</tr>
<tr>
<td><strong>Delete</strong>: Deletes the entries which correspond to the entry flow.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
<tr>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the <strong>[Repository Content]</strong> window.</td>
</tr>
</tbody>
</table>

| Built-in: The schema is created and stored locally for this component only. Related topic: see **Talend Studio User Guide**. |
**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Where clauses for (for UPDATE and DELETE only)</strong></th>
<th>Enter a clause to filter the data to be updated or deleted during the update or delete operations.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default Table Name</strong></td>
<td>Enter the default table name, between double quotation marks.</td>
</tr>
<tr>
<td><strong>Default Schema Name</strong></td>
<td>Enter the default schema name, between double quotation marks.</td>
</tr>
<tr>
<td><strong>Table name from connection name is variable</strong></td>
<td>Select this check box when the name of the connection to this component is set to a variable, such as a context variable.</td>
</tr>
<tr>
<td><strong>Use different table name</strong></td>
<td>Select this check box to define a different output table name, between double quotation marks, in the Table name field which appears.</td>
</tr>
<tr>
<td><strong>Mapping</strong></td>
<td>Specify the metadata mapping file for the database to be used. The metadata mapping file is used for the data type conversion between database and Java. For more information about the metadata mapping, see the related documentation for Type mapping.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **tStatCatcher Statistics**                        | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. <strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer. <strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it. For further information about variables, see <em>Talend Studio User Guide</em>.</th>
</tr>
</thead>
</table>
Usage

Usage rule

**tELTMSSqlOutput** is to be used along with the **tELTMSSqlMap**. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.

**Note:**
Note that the ELT components do not handle actual data flow but only schema information.

<table>
<thead>
<tr>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the <strong>Install</strong> button on the <strong>Component</strong> tab view. You can also find out and add all missing JARs easily on the <strong>Modules</strong> tab in the <strong>Integration</strong> perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

Related scenarios

- **Scenario 1: Aggregating table columns and filtering** on page 744
- **Scenario 2: ELT using an Alias table** on page 748
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTMySQLInput

Adding as many Input tables as required for the most complicated Insert statement.

The three ELT Mysql components are closely related, in terms of their operating conditions. These components should be used to handle Mysql DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

tELTMySQLInput provides the table schema to be used for the SQL statement to execute.

**tELTMySQLInput Standard properties**

These properties are used to configure tELTMySQLInput running in the Standard Job framework.

The Standard tELTMySQLInput component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description, it defines the nature and number of fields to be processed. The schema is either built-in or remotely stored in the Repository. The Schema defined is then passed on to the ELT Mapper to be included to the Insert SQL statement.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

- **Built-in**: The schema is created and stored locally for this component only. Related topic: see [Talend Studio User Guide](#).

- **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see [Talend Studio User Guide](#).

**Default Table Name**

Enter the default table name, between double quotation marks.
**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. |
| | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
| | To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. |
| | For further information about variables, see **Talend Studio User Guide**. |

**Usage**

| Usage rule | **tELTMysqlInput** is to be used along with the **tELTMysqlMap**. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name. |
| | **Note**: Note that the ELT components do not handle actual data flow but only schema information. |

**Related scenarios**

- **Scenario 1: Aggregating table columns and filtering** on page 744
- **Scenario 2: ELT using an Alias table** on page 748
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELT MySqlMap

Uses the tables provided as input to feed the parameter in the built statement. The statement can include inner or outer joins to be implemented between tables or between one table and its aliases.

The three ELT Mysql components are closely related, in terms of their operating conditions. These components should be used to handle Mysql DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

tELT MySqlMap helps to graphically build the SQL statement using the table provided as input.

**tELT MySqlMap Standard properties**

These properties are used to configure tELT MySqlMap running in the Standard Job framework.

The Standard tELT MySqlMap component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td></td>
<td>For an example about how to share a database connection across Job levels, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>ELT MySql Map editor</td>
<td>The ELT Map editor allows you to define the output schema as well as build graphically the SQL statement to be executed. The column names of schema can be different from the column names in the database.</td>
</tr>
<tr>
<td>Style link</td>
<td>Select the way in which links are displayed.</td>
</tr>
<tr>
<td></td>
<td><strong>Auto</strong>: By default, the links between the input and output schemas and the Web service parameters are in the form of curves.</td>
</tr>
<tr>
<td></td>
<td><strong>Bezier curve</strong>: Links between the schema and the Web service parameters are in the form of curve.</td>
</tr>
</tbody>
</table>
### Line
Links between the schema and the Web service parameters are in the form of straight lines. This option slightly optimizes performance.

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the Repository file where Properties are stored. The following fields are pre-filled in using fetched data.</td>
</tr>
</tbody>
</table>

| Host | Database server IP address. |
| Port | Listening port number of DB server. |
| Database | Name of the database. |
| **Username and Password** | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |

### Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

### Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

### Usage

| Usage rule | **tELTmysqlMap** is used along with a **tELTmysqlInput** and **tELTmysqlOutput**. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name. |

| Note: |  |
The ELT components do not handle actual data flow but only schema information.

### Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the Component List box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

---

### Scenario 1: Aggregating table columns and filtering

This scenario describes a Job that gathers together several input DB table schemas and implementing a clause to filter the output using an SQL statement.

![Diagram of Scenario 1: Aggregating table columns and filtering]
• Drop the following components from the Palette onto the design workspace: three tELTMysqlInput components, a tELTMysqlMap, and a tELTMysqlOutput. Label these components to best describe their functionality.

• Double-click the first tELTMysqlInput component to display its Basic settings view.

• Select Repository from the Schema list, click the three dot button preceding Edit schema, and select your DB connection and the desired schema from the [Repository Content] dialog box. The selected schema name appears in the Default Table Name field automatically.

In this use case, the DB connection is Talend_MySQL and the schema for the first input component is owners.

• Set the second and third tELTMysqlInput components in the same way but select cars and resellers respectively as their schema names.

Note: In this use case, all the involved schemas are stored in the Metadata node of the Repository tree view for easy retrieval. For further information concerning metadata, see Talend Studio User Guide.

You can also select the three input components by dropping the relevant schemas from the Metadata area onto the design workspace and double-clicking tELTMysqlInput from the [Components] dialog box. Doing so allows you to skip the steps of labeling the input components and defining their schemas manually.

• Connect the three tELTMysqlInput components to the tELTMysqlMap component using links named following strictly the actual DB table names: owners, cars and resellers.

• Connect the tELTMysqlMap component to the tELTMysqlOutput component and name the link agg_result, which is the name of the database table you will save the aggregation result to.

• Click the tELTMysqlMap component to display its Basic settings view.
• Select **Repository** from the **Property Type** list, and select the same DB connection that you use for the input components.

All the database details are automatically retrieved.

• Leave all the other settings as they are.

• Double-click the **tELTmysqlMap** component to launch the ELT Map editor to set up joins between the input tables and define the output flow.

• Add the input tables by clicking the green plus button at the upper left corner of the ELT Map editor and selecting the relevant table names in the [Add a new alias] dialog box.

• Drop the **ID_Owner** column from the **owners** table to the corresponding column of the **cars** table.

• In the **cars** table, select the **Explicit join** check box in front of the **ID_Owner** column. As the default join type, **INNER JOIN** is displayed on the Join list.

• Drop the **ID_Reseller** column from the **cars** table to the corresponding column of the **resellers** table to set up the second join, and define the join as an inner join in the same way.

• Select the columns to be aggregated into the output table, **agg_result**.

• Drop the **ID_Owner**, **Name**, and **ID_Insurance** columns from the **owners** table to the output table.

• Drop the **Registration**, **Make**, and **Color** columns from the **cars** table to the output table.

• Drop the **Name_Reseller** and **City** columns from the **resellers** table to the output table.

• With the relevant columns selected, the mappings are displayed in yellow and the joins are displayed in dark violet.

• Set up a filter in the output table. Click the **Add filter row** button on top of the output table to display the **Additional clauses** expression field, drop the **City** column from the **resellers** table to the expression field, and complete a WHERE clause that reads **resellers.City = 'Augusta'**.
Click the **Generated SQL Select query** tab to display the corresponding SQL statement.

- Click **OK** to save the ELT Map settings.
- Double-click the **tELT_mysqlOutput** component to display its **Basic settings** view.

Select an action from the **Action on data** list as needed.
• Select **Repository** as the schema type, and define the output schema in the same way as you defined the input schemas. In this use case, select *agg_result* as the output schema, which is the name of the database table used to store the mapping result.

**Note:**

You can also use a built-in output schema and retrieve the schema structure from the preceding component; however, make sure that you specify an existing target table having the same data structure in your database.

• Leave all the other settings as they are.

• Save your Job and press **F6** to launch it.

All selected data is inserted in the *agg_result* table as specified in the SQL statement.

---

### Scenario 2: ELT using an Alias table

This scenario describes a Job that maps information from two input tables and an alias table, serving as a virtual input table, to an output table. The *employees* table contains employees’ IDs, their department numbers, their names, and the IDs of their respective managers. The managers are also considered as employees and hence included in the *employees* table. The *dept* table contains the department information. The alias table retrieves the names of the managers from the *employees* table.

<table>
<thead>
<tr>
<th>ID_Owner</th>
<th>Name</th>
<th>ID_Insurance</th>
<th>Registration</th>
<th>Make</th>
<th>Color</th>
<th>Name_Reseller</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>John ROOSEVELT</td>
<td>64</td>
<td>UDR 821</td>
<td>Peugeot</td>
<td>green</td>
<td>Bill OARFIELD</td>
<td>AUGUSTA</td>
</tr>
<tr>
<td></td>
<td>Andrew CGOLIDGE</td>
<td>104</td>
<td>ZYX 387</td>
<td>EMW</td>
<td>yellow</td>
<td>Ulysses FOLK</td>
<td>AUGUSTA</td>
</tr>
<tr>
<td></td>
<td>Martin ADAMS</td>
<td>167</td>
<td>ZF 348</td>
<td>Audi</td>
<td>blue</td>
<td>Rutherford HARRISON</td>
<td>AUGUSTA</td>
</tr>
<tr>
<td>*</td>
<td>(NULL) (NULL)</td>
<td>(NULL) (NULL)</td>
<td>(NULL) (NULL)</td>
<td>(NULL) (NULL)</td>
<td>(NULL)</td>
<td>(NULL)</td>
<td>(NULL)</td>
</tr>
</tbody>
</table>

---

• Drop two **tELTMysqlInput** components, a **tELTMysqlMap** component, and a **tELTMysqlOutput** component to the design workspace, and label them to best describe their functionality.

• Double-click the first **tELTMysqlInput** component to display its **Basic settings** view.
Select Repository from the Schema list, and define the DB connection and schema by clicking the three dot button preceding Edit schema.

The DB connection is Talend_MySQL and the schema for the first input component is employees.

Note:
In this use case, all the involved schemas are stored in the Metadata node of the Repository tree view for easy retrieval. For further information concerning metadata, see Talend Studio User Guide.

Set the second tELTMysqlInput component in the same way but select dept as its schema.

Double-click the tELTMysqlOutput component to display its Basic settings view.

• Select an action from the Action on data list as needed, Insert in this use case.
• Select Repository as the schema type, and define the output schema in the same way as you defined the input schemas. In this use case, select result as the output schema, which is the name of the database table used to store the mapping result.

The output schema contains all the columns of the input schemas plus a ManagerName column.

• Leave all the other parameters as they are.
• Connect the two tELTMysqlInput components to the tELTMysqlMap component using Link connections named strictly after the actual input table names, employees and dept in this use case.
• Connect the tELTMysqlMap component to the tELTMysqlOutput component using a Link connection. When prompted, click Yes to allow the ELT Mapper to retrieve the output table structure from the output schema.
• Click the tELTMysqlMap component and select the Component tab to display its Basic settings view.
Select **Repository** from the **Property Type** list, and select the same DB connection that you use for the input components.

All the DB connection details are automatically retrieved.

Leave all the other parameters as they are.

Click the three-dot button next to **ELT Mysql Map Editor** or double-click the **tELTMysqlMap** component on the design workspace to launch the ELT Map editor.

With the **tELTMysqlMap** component connected to the output component, the output table is displayed in the output area.

Add the input tables, **employees** and **dept**, in the input area by clicking the green plus button and selecting the relevant table names in the [**Add a new alias**] dialog box.

Create an alias table based on the **employees** table by selecting **employees** from the **Select the table to use** list and typing in **Managers** in the **Type in a valid alias** field in the [**Add a new alias**] dialog box.

Drop the **DeptNo** column from the **employees** table to the **dept** table.

Select the **Explicit join** check box in front of the **DeptNo** column of the **dept** table to set up an inner join.

Drop the **ManagerID** column from the **employees** table to the **ID** column of the **Managers** table.

Select the **Explicit join** check box in front of the **ID** column of the **Managers** table and select **LEFT OUTER JOIN** from the **Join** list to allow the output rows to contain Null values.
- Drop all the columns from the `employees` table to the corresponding columns of the output table.
- Drop the `DeptName` and `Location` columns from the `dept` table to the corresponding columns of the output table.
- Drop the `Name` column from the `Managers` table to the `ManagerName` column of the output table.

- Click on the **Generated SQL Select query** tab to display the SQL query statement to be executed.

```
"SELECT employees.ID, employees.DeptNo, employees.Name, employees.ManagerID, dept.DeptName, dept.Location, Managers.Name
FROM employees INNER JOIN dept ON (dept.DeptNo = employees.DeptNo)
LEFT OUTER JOIN employeesManagers ON (Managers.ID = employees.ManagerID)"
```

- Save your Job and press **F6** to run it.
The output database table `result` contains all the information about the employees, including the names of their respective managers.

<table>
<thead>
<tr>
<th>ID</th>
<th>DeptNo</th>
<th>Name</th>
<th>ManagerID</th>
<th>DeptName</th>
<th>Location</th>
<th>ManagerName</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>Alex</td>
<td>(NULL)</td>
<td>R&amp;D</td>
<td>New York</td>
<td>(NULL)</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>Peter</td>
<td>(NULL)</td>
<td>Accounting</td>
<td>Dallas</td>
<td>(NULL)</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>Mark</td>
<td>1</td>
<td>R&amp;D</td>
<td>New York</td>
<td>Alex</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>Michael</td>
<td>1</td>
<td>R&amp;D</td>
<td>New York</td>
<td>Alex</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>Christophe</td>
<td>2</td>
<td>Accounting</td>
<td>Dallas</td>
<td>Peter</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>Stephane</td>
<td>3</td>
<td>R&amp;D</td>
<td>New York</td>
<td>Mark</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>Cedric</td>
<td>3</td>
<td>R&amp;D</td>
<td>New York</td>
<td>Mark</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>Bill</td>
<td>4</td>
<td>R&amp;D</td>
<td>New York</td>
<td>Michael</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>Jack</td>
<td>2</td>
<td>Accounting</td>
<td>Dallas</td>
<td>Peter</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>Andrews</td>
<td>4</td>
<td>R&amp;D</td>
<td>New York</td>
<td>Michael</td>
</tr>
</tbody>
</table>

* NULL) (NULL) (NULL) (NULL) (NULL) (NULL)

**Related scenarios**

- **Scenario: Mapping data using a subquery** on page 800, a related scenario using subquery
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
**tELTMySQLOutput**

**tELTMySQLOutput** executes the SQL Insert, Update and Delete statement to the Mysql database.

The three ELT Mysql components are closely related, in terms of their operating conditions. These components should be used to handle Mysql DB schemas to generate Insert statements, including clauses, which are to be executed in the DB output table defined.

**tELTMySQLOutput** carries out the action on the table specified and inserts the data according to the output schema defined the ELT Mapper.

**tELTMySQLOutput Standard properties**

These properties are used to configure **tELTMySQLOutput** running in the Standard Job framework.

The Standard **tELTMySQLOutput** component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

| Action on data | On the data of the table defined, you can perform the following operation:
|                | **Insert**: Add new entries to the table. If duplicates are found, Job stops.
|                | **Update**: Updates entries in the table.
|                | **Delete**: Deletes the entries which correspond to the entry flow. |

**Schema and Edit schema**

A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either built-in or remotely stored in the Repository.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see **Talend Studio User Guide**.
**Repository:** The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

| Where clauses for (for UPDATE and DELETE only) | Enter a clause to filter the data to be updated or deleted during the update or delete operations. |
| Default Table Name | Enter the default table name, between inverted commas. Note that the table must exist already. If it does not exist, you can use `tCreateTable` to create one first. For more information about `tCreateTable`, see `tCreateTable` on page 541. |
| Table name from connection name is variable | Select this check box when the name of the connection to this component is set to a variable, such as a context variable. |
| Use different table name | Select this check box to define a different output table name, between double quotation marks, in the **Table name** field which appears. |
| Mapping | Specify the metadata mapping file for the database to be used. The metadata mapping file is used for the data type conversion between database and Java. For more information about the metadata mapping, see the related documentation for Type mapping. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | **NB_LINE:** the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**NB_LINE_INSERTED:** the number of rows inserted. This is an After variable and it returns an integer.  
**ERROR-message:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tELTMysqlOutput is to be used along with the tELTMysqlMap. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note:</td>
</tr>
<tr>
<td></td>
<td>Note that the ELT components do not handle actual data flow but only schema information.</td>
</tr>
</tbody>
</table>

**Related scenarios**

- Scenario 1: Aggregating table columns and filtering on page 744
- Scenario 2: ELT using an Alias table on page 748
- Aggregating Snowflake data using context variables as table and connection names on page 724
tELTNetezzaInput

Allows you to add as many Input tables as required for the most complicated Insert statement.

The three ELT Netezza components are closely related, in terms of their operating conditions. These components should be used to handle Netezza database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

Provides the table schema to be used for the SQL statement to execute.

tELTNetezzaInput Standard properties

These properties are used to configure tELTNetezzaInput running in the Standard Job framework.

The Standard tELTNetezzaInput component belongs to the ELT family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only. • <strong>Change to built-in property</strong>: choose this option to change the schema to Built-in for local changes. • <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td>Built-in: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Repository: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Default Table Name</td>
<td>Type in the default table name.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |
Global Variables

Global Variables

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

Usage rule

**tELTNetezzaInput** is to be used along with the **tELTNetezzaMap**. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.

**Note:**

Note that the ELT components do not handle actual data flow but only schema information.

Related scenarios

- **Scenario: Mapping data using a simple implicit join** on page 682
- **Scenario 1: Aggregating table columns and filtering** on page 744
- **Scenario 2: ELT using an Alias table** on page 748
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTNetezzaMap

Uses the tables provided as input, to feed the parameter in the built statement. The statement can include inner or outer joins to be implemented between tables or between one table and its aliases.

The three ELT Netezza components are closely related, in terms of their operating conditions. These components should be used to handle Netezza database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

Helps you to build the SQL statement graphically, using the table provided as input.

tELTNetezzaMap Standard properties

These properties are used to configure tELTNetezzaMap running in the Standard Job framework.

The Standard tELTNetezzaMap component belongs to the ELT family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td>ELT Netezza Map Editor</td>
<td>The ELT Map editor allows you to define the output schema and make a graphical build of the SQL statement to be executed. The column names of schema can be different from the column names in the database.</td>
</tr>
</tbody>
</table>
| Style link                | Select the way in which links are displayed.  
**Auto:** By default, the links between the input and output schemas and the Web service parameters are in the form of curves.  
**Bezier curve:** Links between the schema and the Web service parameters are in the form of curve. |
**Line (fastest):** Links between the schema and the Web service parameters are in the form of straight lines. This option slightly optimizes performance.

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in:</strong> No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> Select the Repository file where Properties are stored. The following fields are filled in using fetched data.</td>
</tr>
</tbody>
</table>

- **Host**
  - Database server IP address.

- **Port**
  - Listening port number of DB server.

- **Database**
  - Name of the database.

- **Username and Password**
  - DB user authentication data.
  - To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

**Advanced settings**

- **Additional JDBC parameters**
  - Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the **Basic settings**.

- **tStatCatcher Statistics**
  - Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

**Global Variables**

- **Global Variables**
  - **ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.
  
  A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

  To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

  For further information about variables, see *Talend Studio User Guide.*
### Usage

#### Usage rule

**tELTNetezzaMap** is used along with **tELTNetezzaInput** and **tELTNetezzaOutput**. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.

**Note:**

Note that the ELT components do not handle actual data flow but only schema information.

#### Dynamic settings

Click the [*] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Related scenarios

- **Scenario: Mapping data using a simple implicit join** on page 682
- **Scenario 1: Aggregating table columns and filtering** on page 744
- **Scenario 2: ELT using an Alias table** on page 748
- **Scenario: Mapping data using a subquery** on page 800, a related scenario using subquery
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
**tELTNetezzaOutput**

Performs the action (insert, update or delete) on data in the specified Netezza table through the SQL statement generated by the tELTNetezzaMap component.

The three ELT Netezza components are closely related, in terms of their operating conditions. These components should be used to handle Netezza database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

**tELTNetezzaOutput Standard properties**

These properties are used to configure tELTNetezzaOutput running in the Standard Job framework. The Standard tELTNetezzaOutput component belongs to the ELT family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Action on data</th>
<th>On the data of the table defined, you can perform the following operation:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Insert</strong>: Adds new entries to the table.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Updates entries in the table.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Deletes the entries which correspond to the entry flow.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description, it defines the number of fields that will be processed and passed on to the next component.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Sync columns</strong> to retrieve the schema from the previous component connected in the Job.</td>
</tr>
</tbody>
</table>

| Built-in | The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*. |
**Repository:** The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide.*

<table>
<thead>
<tr>
<th><strong>Where clauses for (for UPDATE and DELETE only)</strong></th>
<th>Enter a clause to filter the data to be updated or deleted during the update or delete operations.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default Table Name</strong></td>
<td>Enter the default table name, between double quotation marks.</td>
</tr>
<tr>
<td><strong>Table name from connection name is variable</strong></td>
<td>Select this check box when the name of the connection to this component is set to a variable, such as a context variable.</td>
</tr>
<tr>
<td><strong>Use different table name</strong></td>
<td>Select this check box to define a different output table name, between double quotation marks, in the <strong>Table name</strong> field that appears.</td>
</tr>
<tr>
<td><strong>Mapping</strong></td>
<td>Specify the metadata mapping file for the database to be used. The metadata mapping file is used for the data type conversion between database and Java. For more information about the metadata mapping, see the related documentation for Type mapping.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

| **Global Variables** | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide.* |

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tELTNetezzaOutput is to be used along with the tELTNetezzaMap. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note: Note that the ELT components do not handle actual data flow but only schema information.</td>
</tr>
</tbody>
</table>

Related scenarios

- Scenario: Mapping data using a simple implicit join on page 682
- Scenario 1: Aggregating table columns and filtering on page 744
- Scenario 2: ELT using an Alias table on page 748
- Aggregating Snowflake data using context variables as table and connection names on page 724
tELTOracleInput

Provides the Oracle table schema that will be used by the tELTOracleMap component to generate the SQL SELECT statement.

The three ELT Oracle components are closely related, in terms of their operating conditions. These components should be used to handle Oracle database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

tELTOracleInput Standard properties

These properties are used to configure tELTOracleInput running in the Standard Job framework.

The Standard tELTOracleInput component belongs to the ELT family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description, it defines the nature and number of fields to be processed. The schema is either built-in or remotely stored in the Repository. The Schema defined is then passed on to the ELT Mapper to be included to the Insert SQL statement. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td>Built-in: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Repository: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Default Table Name</td>
<td>Enter the default table name, between double quotation marks.</td>
</tr>
<tr>
<td>Default Schema Name</td>
<td>Enter the default schema name, between double quotation marks.</td>
</tr>
</tbody>
</table>
Advanced settings

| tStatCatcher Statistics                  | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tELTOracleInput is to be used along with the tELTOracleMap. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note: The ELT components do not handle actual data flow but only schema information.</td>
</tr>
</tbody>
</table>

Related scenarios

- Scenario: Updating Oracle DB entries on page 768
- Aggregating Snowflake data using context variables as table and connection names on page 724
tELTOracleMap

Builds the SQL SELECT statement using the table schema(s) provided by one or more tELTOracleInput components.

The three ELT Oracle components are closely related, in terms of their operating conditions. These components should be used to handle Oracle database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

tELTOracleMap Standard properties

These properties are used to configure tELTOracleMap running in the Standard Job framework.
The Standard tELTOracleMap component belongs to the ELT family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td></td>
<td>For an example about how to share a database connection across Job levels, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

| ELT Oracle Map Editor     | The ELT Map editor allows you to define the output schema and make a graphical build of the SQL statement to be executed. The column names of schema can be different from the column names in the database. |

<table>
<thead>
<tr>
<th>Style link</th>
<th><strong>Auto:</strong> By default, the links between the input and output schemas and the Web service parameters are in the form of curves.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Bezier curve:</strong> Links between the schema and the Web service parameters are in the form of curve.</td>
</tr>
<tr>
<td></td>
<td><strong>Line:</strong> Links between the schema and the Web service parameters are in the form of straight lines.</td>
</tr>
</tbody>
</table>
This option slightly optimizes performance.

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the Repository file where Properties are stored. The following fields are pre-filled in using fetched data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connection type</th>
<th>Drop-down list of the available drivers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Version</td>
<td>Select the Oracle version you are using.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>

| Mapping                | Automatically set mapping parameter. |

### Advanced settings

<table>
<thead>
<tr>
<th>Additional JDBC Parameters</th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <strong>Use an existing connection</strong> check box in the <strong>Basic settings</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Hint Options</td>
<td>Select this check box to activate the hint configuration area to help you optimize a query's execution. In this area, parameters are: - <strong>HINT</strong>: specify the hint you need, using the syntax /* + */. - <strong>POSITION</strong>: specify where you put the hint in a SQL statement. - <strong>SQL STMT</strong>: select the SQL statement you need to use.</td>
</tr>
</tbody>
</table>

| tStatCatcher Statistics   | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</th>
</tr>
</thead>
</table>
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

**Usage**

**Usage rule**

tELTOracleMap is used along with a tELTOracleInput and tELTOracleOutput. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.

Note:

Note that the ELT components do not handle actual data flow but only schema information.

**Dynamic settings**

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

**Scenario: Updating Oracle DB entries**

This scenario is based on the data aggregation scenario, Scenario 1: Aggregating table columns and filtering on page 744. As the data update action is available in Oracle DB, this scenario describes a Job that updates particular data in the agg_result table.
• As described in Scenario 1: Aggregating table columns and filtering on page 744, set up a Job for data aggregation using the corresponding ELT components for Oracle DB, \texttt{tELTOracleInput}, \texttt{tELTOracleMap}, and \texttt{tELTOracleOutput}, and execute the Job to save the aggregation result in a database table named \textit{agg\_result}.

\textbf{Note:}

When defining filters in the ELT Map editor, note that strings are case sensitive in Oracle DB.

• Launch the ELT Map editor and add a new output table named \textit{update\_data}.

• Add a filter row to the \textit{update\_data} table to set up a relationship between input and output tables:
  \begin{align*}
  \text{owners.ID\_OWNER} &= \text{agg\_result.ID\_OWNER}.
  \end{align*}

• Drop the \textit{MAKE} column from the \textit{cars} table to the \textit{update\_data} table.

• Drop the \textit{NAME\_RESELLER} column from the \textit{resellers} table to the \textit{update\_data} table.

• Add a model enclosed in single quotation marks, \textit{A8} in this use case, to the \textit{MAKE} column from the \textit{cars} table, preceded by a double pipe.

• Add \textit{Sold by} enclosed in single quotation marks in front of the \textit{NAME\_RESELLER} column from the \textit{resellers} table, with a double pipe in between.

• Check the \textbf{Generated SQL select query} tab to be executed.
Click OK to validate the changes in the ELT Mapper.

Deactivate the tELTOracleOutput component labeled Agg_Result by right-clicking it and selecting Deactivate Agg_Result from the contextual menu.

Drop a new tELTOracleOutput component from the Palette to the design workspace, and label it Update_Data to better identify its functionality.

Connect the tELTOracleMap component to the new tELTOracleOutput component using the link corresponding to the new output table defined in the ELT Mapper, update_data in this use case.

Double-click the new tELTOracleOutput component to display its Basic settings view.

From the Action on data list, select Update.

Check the schema, and click Sync columns to retrieve the schema structure from the preceding component if necessary.

In the WHERE clauses area, add a clause that reads agg_result.MAKE = 'Audi' to update data relating to the make of Audi in the database table agg_result.

Fill the Default Table Name field with the name of the output link, update_data in this use case.

Select the Use different table name check box, and fill the Table name field with the name of the database table to be updated, agg_result in this use case.

Leave the other parameters as they are.

Save your Job and press F6 to run it.

The relevant data in the database table is updated as defined.
<table>
<thead>
<tr>
<th>ID_OWNER</th>
<th>NAME</th>
<th>ID_INS</th>
<th>REGISTR...</th>
<th>MAKE</th>
<th>COLOR</th>
<th>NAME_RESELLER</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Martin ADAMS</td>
<td>167</td>
<td>ZFF 348</td>
<td>Audi A8</td>
<td>grey</td>
<td>Sold by Rutherford HARRISON</td>
<td>AUGUSTA</td>
</tr>
<tr>
<td>11</td>
<td>Andrew COOLIDGE</td>
<td>104</td>
<td>ZYX 307</td>
<td>BMW</td>
<td>yellow</td>
<td>Ulysses POLK</td>
<td>AUGUSTA</td>
</tr>
<tr>
<td>69</td>
<td>John ROOSEVELT</td>
<td>84</td>
<td>UCR 821</td>
<td>Peugeot</td>
<td>green</td>
<td>Bill GARFIELD</td>
<td>AUGUSTA</td>
</tr>
</tbody>
</table>

**Related scenario**

- **Scenario: Updating Oracle DB entries** on page 768
- **Scenario: Mapping data using a subquery** on page 800, a related scenario using subquery
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
**tELTOracleOutput**

Performs the action (insert, update, delete, or merge) on data in the specified Oracle table through the SQL statement generated by the tELTOracleMap component.

The three ELT Oracle components are closely related, in terms of their operating conditions. These components should be used to handle Oracle database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

**tELTOracleOutput Standard properties**

These properties are used to configure tELTOracleOutput running in the Standard Job framework.

The Standard tELTOracleOutput component belongs to the ELT family.

The component in this framework is generally available.

**Basic Settings**

| Action on data | On the data of the table defined, you can perform the following operation:
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Insert</strong></td>
<td>Add new entries to the table. If duplicates are found, the Job stops.</td>
</tr>
<tr>
<td><strong>Update</strong></td>
<td>Updates entries in the table.</td>
</tr>
<tr>
<td><strong>Delete</strong></td>
<td>Deletes the entries which correspond to the entry flow.</td>
</tr>
<tr>
<td><strong>MERGE</strong></td>
<td>Updates and/or adds data to the table. Note that the options available for the MERGE operation are different to those available for the Insert, Update or Delete operations.</td>
</tr>
</tbody>
</table>

**Note:**

Following global variables are available:

- **NB_LINE_INSERTED**: Number of lines inserted during the **Insert** operation.
- **NB_LINE_UPDATED**: Number of lines updated during the **Update** operation.
- **NB_LINE_DELETED**: Number of lines deleted during the **Delete** operation.
- **NB_LINE_MERGED**: Number of lines inserted and/or updated during the **MERGE** operation.

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either built-in or remotely stored in the Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
<td></td>
</tr>
</tbody>
</table>
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>The schema is created and stored locally for this component only. Related topic: see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

**Where clauses for (for UPDATE and DELETE only)**
Enter a clause to filter the data to be updated or deleted during the update or delete operations.

**Use Merge Update (for MERGE)**
Select this check box to update the data in the output table.
- **Column**: Lists the columns in the entry flow.
- **Update**: Select the check box which corresponds to the name of the column you want to update.
- **Use Merge Update Where Clause**: Select this check box and enter the WHERE clause required to filter the data to be updated, if necessary.
- **Use Merge Update Delete Clause**: Select this check box and enter the WHERE clause required to filter the data to be deleted and updated, if necessary.

**Use Merge Insert (for MERGE)**
Select this check box to insert the data in the table.
- **Column**: Lists the entry flow columns.
- **Check All**: Select the check box corresponding to the name of the column you want to insert.
- **Use Merge Update Where Clause**: Select this check box and enter the WHERE clause required to filter the data to be inserted.

**Default Table Name**
Enter a default name for the table, between double quotation marks.

**Default Schema Name**
Enter a name for the default Oracle schema, between double quotation marks.

**Table name from connection name is variable**
Select this check box when the name of the connection to this component is set to a variable, such as a context variable.
<table>
<thead>
<tr>
<th>Use different table name</th>
<th>Select this check box to define a different output table name, between double quotation marks, in the <strong>Table name</strong> field which appears.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapping</td>
<td>Specify the metadata mapping file for the database to be used. The metadata mapping file is used for the data type conversion between database and Java. For more information about the metadata mapping, see the related documentation for Type mapping.</td>
</tr>
<tr>
<td><strong>Advanced settings</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Use Hint Options</strong></td>
<td>Select this check box to activate the hint configuration area when you want to use a hint to optimize a query's execution. In this area, parameters are:</td>
</tr>
<tr>
<td></td>
<td>- <strong>HINT</strong>: specify the hint you need, using the syntax /*+ */.</td>
</tr>
<tr>
<td></td>
<td>- <strong>POSITION</strong>: specify where you put the hint in a SQL statement.</td>
</tr>
<tr>
<td></td>
<td>- <strong>SQL STMT</strong>: select the SQL statement you need to use.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
<tr>
<td><strong>Global Variables</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Global Variables</strong></td>
<td><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td><strong>Usage</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Usage rule</strong></td>
<td><strong>tELTOracleOutput</strong> is to be used along with the <strong>tELTOracleInput</strong> and <strong>tELTOracleMap</strong> components. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.</td>
</tr>
</tbody>
</table>
Scenario: Using the Oracle MERGE function to update and add data simultaneously

This scenario describes a Job that allows you to add new customer information and update existing customer information in a database table using the Oracle MERGE command.

Linking the components

Procedure
1. Drop the following components from the Palette to the design workspace: `tELTOracleInput`, `tELTOracleMap`, and `tELTOracleOutput`.
2. Label `tELTOracleInput` as `new_customer`, `tELTOracleMap` as `ELT Mapper`, and `tELTOracleOutput` as `merge_data`.
3. Link `tELTOracleInput` to `tELTOracleMap` using a `Row > New Output (table)` connection. In the pop-up box, enter `NEW_CUSTOMERS` as the table name, which should be the actual database table name.
4. Link `tELTOracleMap` to `tELTOracleOutput` using a `Row > New Output (table)` connection. In the pop-up box, enter `customers_merge` as the name of the database table, which holds the merge results.

Configuring the components

Procedure
1. Double-click the `tELTOracleInput` component to display its Basic settings view.

   ![Diagram](image)

   **tELTOracleInput**

   - **new_customer**
   - **NEW_CUSTOMERS (Table)**
   - **ELT Mapper**
   - **customers_merge (Table)**
   - **merge_data**

2. Select **Repository** from the **Schema** list, click the [...] button preceding **Edit schema**, and select your database connection and the desired schema from the [Repository Content] dialog box.
The selected schema name appears in the **Default Table Name** field automatically. In this use case, the database connection is `Talend_Oracle` and the schema is `new_customers`.

**Note:**

In this use case, the input schema is stored in the **Metadata** node of the **Repository** tree view for easy retrieval. For further information concerning metadata, see *Talend Studio User Guide*.

You can also select the input component by dropping the relevant schema from the **Metadata** area onto the design workspace and double-clicking `tELTOracleInput` from the **Components** dialog box. Doing so allows you to skip the steps of labeling the input component and defining its schema manually.

3. Click the `tELTOracleMap` component to display its **Basic settings** view.

4. Select **Repository** from the **Property Type** list, and select the same database connection that you use for the input components.

   All the database details are automatically retrieved.

   Leave the other settings as they are.

5. Double-click the `tELTOracleMap` component to launch the ELT Map editor to set up the data transformation flow.

   Display the input table by clicking the green plus button at the upper left corner of the ELT Map editor and selecting the relevant table name in the **Add a new alias** dialog box.

   In this use case, the only input table is `new_customers`. 
6. Select all the columns in the input table and drop them to the output table.

7. Click the **Generated SQL Select query tab** to display the query statement to be executed.

Click **OK** to validate the ELT Map settings and close the ELT Map editor.

8. Double-click the **tELTOracleOutput** component to display its **Basic settings** view.

   From the **Action on data** list, select **MERGE**.

   Click the **Sync columns** button to retrieve the schema from the preceding component.

   Select the **Use Merge Update** check box to update the data using Oracle's MERGE function.

9. In the table that appears, select the check boxes for the columns you want to update.

   In this use case, we want to update all the data according to the customer ID. Therefore, select all the check boxes except the one for the **ID** column.
Warning: The columns defined as the primary key cannot and must not be made subject to updates.

10. Select the Use Merge Insert check box to insert new data while updating the existing data by leveraging Oracle’s MERGE function.

In the table that appears, select the check boxes for the columns into which you want to insert new data.

In this use case, we want to insert all the new customer data. Therefore, select all the check boxes by clicking the Check All check box.

11. Fill the Default Table Name field with the name of the target table already existing in your database. In this example, fill in customers_merge.

12. Leave the other parameters as they are.
Executing the Job

Procedure

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to run the Job.

The data is updated and inserted in the database. The query used is displayed on the console.

```sql
[statistics] connecting to socket on port 3712
[statistics] connected
Merge with:
MERGE INTO customers_merge target USING (SELECT NEW_CUSTOMERS.ID ,
NEW_CUSTOMERS.FIRSTNAME , NEW_CUSTOMERS.LASTNAME , NEW_CUSTOMERS.ADDRESS ,
NEW_CUSTOMERS.REGISTERDATE , NEW_CUSTOMERS.REVENUE , NEW_CUSTOMERS.STATE FROM
NEW_CUSTOMERS) source ON (target.ID=source.ID) WHEN MATCHED THEN UPDATE SET
  target.FIRSTNAME=source.FIRSTNAME , target.LASTNAME=source.LASTNAME , target.ADDRESS=
source.ADDRESS , target.REGISTERDATE=source.REGISTERDATE , target.REVENUE=source.REVENUE ,
target.STATE=source.STATE WHEN NOT MATCHED THEN INSERT (ID, FIRSTNAME, LASTNAME, ADDRESS, REGISTERDATE, REVENUE, STATE) VALUES (source.ID, source.FIRSTNAME, source.LASTNAME, source.ADDRESS, source.REGISTERDATE, source.REVENUE, source.STATE)
Merge with:
MERGE INTO customers_merge target USING (SELECT NEW_CUSTOMERS.ID ,
NEW_CUSTOMERS.FIRSTNAME , NEW_CUSTOMERS.LASTNAME , NEW_CUSTOMERS.ADDRESS ,
NEW_CUSTOMERS.REGISTERDATE , NEW_CUSTOMERS.REVENUE , NEW_CUSTOMERS.STATE FROM
NEW_CUSTOMERS) source ON (target.ID=source.ID) WHEN MATCHED THEN UPDATE SET
  target.FIRSTNAME=source.FIRSTNAME , target.LASTNAME=source.LASTNAME , target.ADDRESS=
source.ADDRESS , target.REGISTERDATE=source.REGISTERDATE , target.REVENUE=source.REVENUE ,
target.STATE=source.STATE WHEN NOT MATCHED THEN INSERT (ID, FIRSTNAME, LASTNAME, ADDRESS, REGISTERDATE, REVENUE, STATE) VALUES (source.ID, source.FIRSTNAME, source.LASTNAME, source.ADDRESS, source.REGISTERDATE, source.REVENUE, source.STATE)
[statistics] disconnected
```
tELTPostgresqlInput

Provides the Postgresql table schema that will be used by the tELTPostgresqlMap component to generate the SQL SELECT statement.

The three ELT Postgresql components are closely related, in terms of their operating conditions. These components should be used to handle Postgresql database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

tELTPostgresqlInput Standard properties

These properties are used to configure tELTPostgresqlInput running in the Standard Job framework. The Standard tELTPostgresqlInput component belongs to the ELT family. The component in this framework is generally available.

Basic settings

| Schema and Edit schema | A schema is a row description, it defines the nature and number of fields to be processed. The schema is either built-in or remotely stored in the Repository. The Schema defined is then passed on to the ELT Mapper to be included to the Insert SQL statement. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
- View schema: choose this option to view the schema only.  
- Change to built-in property: choose this option to change the schema to Built-in for local changes.  
- Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in:</td>
<td>The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Repository:</td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Default Table Name</td>
<td>Enter the default table name, between double quotation marks.</td>
</tr>
<tr>
<td>Default Schema Name</td>
<td>Enter the default schema name, between double quotation marks.</td>
</tr>
</tbody>
</table>
Advanced settings

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**

**tELTPostgresqlInput** is to be used along with the **tELTPostgresqlMap**. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.

**Note:**

Note that the ELT components do not handle actual data flow but only schema information.

Related scenarios

- **Scenario 1: Aggregating table columns and filtering** on page 744
- **Scenario 2: ELT using an Alias table** on page 748
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTPostgresqlMap

Builds the SQL SELECT statement using the table schema(s) provided by one or more tELTPostgresql Input components.

The three ELT Postgresql components are closely related, in terms of their operating conditions. These components should be used to handle Postgresql database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

**tELTPostgresqlMap Standard properties**

These properties are used to configure tELTPostgresqlMap running in the Standard Job framework. The Standard tELTPostgresqlMap component belongs to the ELT family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong> \nWhen a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to: \n1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection. \n2. In the child level, use a dedicated connection component to read that registered database connection. \nFor an example about how to share a database connection across Job levels, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ELT Postgresql Map Editor</th>
<th>The ELT Map editor allows you to define the output schema and make a graphical build of the SQL statement to be executed. The column names of schema can be different from the column names in the database.</th>
</tr>
</thead>
</table>

| Style link                  | Select the way in which links are displayed. \n**Auto:** By default, the links between the input and output schemas and the Web service parameters are in the form of curves. \n**Bezier curve:** Links between the schema and the Web service parameters are in the form of curve. |
**Line**: Links between the schema and the Web service parameters are in the form of straight lines. This option slightly optimizes performance.

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>:</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong>:</td>
<td>Select the Repository file where Properties are stored. The following fields are pre-filled in using fetched data.</td>
</tr>
</tbody>
</table>

| Host | Database server IP address |
| Port | Listening port number of DB server. |
| Database | Name of the database |

**Username and Password**
DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

### Advanced settings

**Additional JDBC parameters**
Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the Basic settings.

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

### Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.
Usage

Usage rule

<table>
<thead>
<tr>
<th>tELTPostgresqlMap is used along with a tELTPostgresqlInput and tELTPostgresqlOutput. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note: Note that the ELT components do not handle actual data flow but only schema information.</td>
</tr>
</tbody>
</table>

Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

- Scenario 1: Aggregating table columns and filtering on page 744
- Scenario 2: ELT using an Alias table on page 748
- Scenario: Mapping data using a subquery on page 800, a related scenario using subquery
- Aggregating Snowflake data using context variables as table and connection names on page 724
tELTPostgresqlOutput

Performs the action (insert, update or delete) on data in the specified Postgresql table through the SQL statement generated by the tELTPostgresqlMap component.

The three ELT Postgresql components are closely related, in terms of their operating conditions. These components should be used to handle Postgresql database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

tELTPostgresqlOutput Standard properties

These properties are used to configure tELTPostgresqlOutput running in the Standard Job framework. The Standard tELTPostgresqlOutput component belongs to the ELT family. The component in this framework is generally available.

Basic settings

| Action on data | On the data of the table defined, you can perform the following operation:
|               | **Insert**: Add new entries to the table. If duplicates are found, Job stops.
|               | **Update**: Updates entries in the table.
|               | **Delete**: Deletes the entries which correspond to the entry flow. |

Schema and Edit schema

A schema is a row description, that is to say, it defines the number of fields to be processed and passed on to the next component. The schema is either built-in or remotely stored in the Repository.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*. 


Repository: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

**Where clauses for (for UPDATE and DELETE only)**

Enter a clause to filter the data to be updated or deleted during the update or delete operations.

**Default Table Name**

Enter the default table name between double quotation marks.

**Default Schema Name**

Enter the default schema name between double quotation marks.

**Table name from connection name is variable**

Select this check box when the name of the connection to this component is set to a variable, such as a context variable.

**Use different table name**

Select this check box to enter a different output table name, between double quotation marks, in the Table name field which appears.

**Mapping**

Specify the metadata mapping file for the database to be used. The metadata mapping file is used for the data type conversion between database and Java. For more information about the metadata mapping, see the related documentation for Type mapping.

**Advanced settings**

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

**Global Variables**

**Global Variables**

| NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. |
| NB_LINE_INSERTED: the number of rows inserted. This is an After variable and it returns an integer. |
| ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press *Ctrl + Space* to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*. |
## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th><strong>tELTPostgresqlOutput</strong> is to be used along with the <strong>tELTPostgresqlMap</strong>. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>Note that the ELT components do not handle actual data flow but only schema information.</td>
</tr>
</tbody>
</table>

## Related scenarios

- **Scenario 1:** Aggregating table columns and filtering on page 744
- **Scenario 2:** ELT using an Alias table on page 748
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
**tELTSybaseInput**

Provides the Sybase table schema that will be used by the tELTSybaseMap component to generate the SQL SELECT statement.

The three ELT Sybase components are closely related, in terms of their operating conditions. These components should be used to handle Sybase database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

**tELTSybaseInput Standard properties**

These properties are used to configure tELTSybaseInput running in the Standard Job framework. The Standard tELTSybaseInput component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

| **Schema and Edit schema** | A schema is a row description, it defines the number and nature of the fields to be processed. The schema is either built-in (local) or stored remotely in the Repository. The Schema defined is then passed on to the ELT Mapper for inclusion in the Insert SQL statement. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  • **View schema**: choose this option to view the schema only.  
  • **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.  

| **Built-in**: The schema is created and stored locally for this component only. Related topic: see **Talend Studio User Guide**. |
| **Repository**: The schema already exists and is stored in the Repository. Hence, it can be re-used for other projects and Jobs. Related topic: see **Talend Studio User Guide**. |
| **Default Table Name** | Enter a default name for the table, between double quotation marks. |
| **Default Schema Name** | Enter a default name for the Sybase schema, between double quotation marks. |
Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | tELTSybaseInput is intended for use with tELTSybaseMap. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name. Note: ELT components only handle schema information. They do not handle actual data flow. |

| Limitation | This component requires installation of its related jar files. |

Related scenarios

- Scenario 1: Aggregating table columns and filtering on page 744
- Scenario 2: ELT using an Alias table on page 748
- Aggregating Snowflake data using context variables as table and connection names on page 724
tELTSybaseMap

Builds the SQL SELECT statement using the table schema(s) provided by one or more tELTSybaseInput components.

The three ELT Sybase components are closely related, in terms of their operating conditions. These components should be used to handle Sybase database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

**tELTSybaseMap Standard properties**

These properties are used to configure tELTSybaseMap running in the Standard Job framework. The Standard tELTSybaseMap component belongs to the ELT family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Use an existing connection</strong></th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td></td>
<td>For an example about how to share a database connection across Job levels, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

| **ELT Sybase Map Editor** | The ELT Map editor allows you to define the output schema and make a graphical build of the SQL statement to be executed. The column names of schema can be different from the column names in the database. |

<table>
<thead>
<tr>
<th><strong>Style link</strong></th>
<th>Select the way in which links are displayed.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Auto:</strong> By default, the links between the input and output schemas and the Web service parameters are in the form of curves.</td>
</tr>
<tr>
<td></td>
<td><strong>Bezier curve:</strong> Links between the schema and the Web service parameters are in the form of curve.</td>
</tr>
</tbody>
</table>
**Line:** Links between the schema and the Web service parameters are in the form of straight lines. This option slightly optimizes performance.

<table>
<thead>
<tr>
<th>Property type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data is stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the Repository file where the component properties are stored. The following fields are pre-filled using collected data.</td>
</tr>
</tbody>
</table>

**DB Version**
Select the version of the Sybase database to be used from the drop-down list.

**Host**
Database server IP address.

**Port**
Listening port number of DB server.

**Database**
Name of the database.

**Username and Password**
DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**Advanced settings**
- **tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at a Job level as well as at component level.

**Global Variables**
- **Global Variables**
  - **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.
  - A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
  - To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
  - For further information about variables, see *Talend Studio User Guide*.

**Usage**
- **Usage rule**
tELTSybaseMap is intended for use with tELTSybaseInput and tELTSybaseOutput. Note that the Output link to be
used with these components must correspond strictly to the syntax of the table name.

**Note:**
The ELT components only handle schema information. They do not handle actual data flow.

### Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see [Scenario: Reading data from databases through context-based dynamic connections](#) on page 2497 and [Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters](#) on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Limitation

This component requires installation of its related jar files.

## Related scenarios

- [Scenario 1: Aggregating table columns and filtering](#) on page 744
- [Scenario 2: ELT using an Alias table](#) on page 748
- [Aggregating Snowflake data using context variables as table and connection names](#) on page 724
tELTSybaseOutput

Performs the action (insert, update or delete) on data in the specified Sybase table through the SQL statement generated by the tELTSybaseMap component.

The three ELT Sybase components are closely related, in terms of their operating conditions. These components should be used to handle Sybase database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

tELTSybaseOutput Standard properties

These properties are used to configure tELTSybaseOutput running in the Standard Job framework. The Standard tELTSybaseOutput component belongs to the ELT family. The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Action on data</th>
<th>On the data of the table defined, you can perform the following operation:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, the Job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Updates entries in the table.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Deletes the entries which correspond to the entry flow.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description, that is to say, it defines the number and nature of the fields to be processed and passed on to the next component. The schema is either <strong>Built-in</strong> (local) or stored remotely in the Repository. The Schema defined is then passed on to the ELT Mapper for inclusion in the Insert SQL statement.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
Repository: The schema already exists and is stored in the Repository. Hence, it can be re-used for other projects and Jobs. Related topic: see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Where clauses for (for UPDATE and DELETE only)</th>
<th>Enter a clause to filter the data to be updated or deleted during the update or delete operations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Table Name</td>
<td>Enter a default name for the table, between double quotation marks. Note that the table must exist already. If it does not exist, you can use <code>tCreateTable</code> to create one first. For more information about <code>tCreateTable</code>, see <code>tCreateTable</code> on page 541.</td>
</tr>
<tr>
<td>Default Schema Name</td>
<td>Enter a default name for the Sybase schema, between double quotation marks.</td>
</tr>
<tr>
<td>Table name from connection name is variable</td>
<td>Select this check box when the name of the connection to this component is set to a variable, such as a context variable.</td>
</tr>
<tr>
<td>Use different table name</td>
<td>Select this check box to enter a different output table name, between double quotation marks, in the Table name field which appears.</td>
</tr>
<tr>
<td>Mapping</td>
<td>Specify the metadata mapping file for the database to be used. The metadata mapping file is used for the data type conversion between database and Java. For more information about the metadata mapping, see the related documentation for Type mapping.</td>
</tr>
</tbody>
</table>

**Advanced settings**

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at a Job level as well as at component level.

**Global Variables**

**Global Variables**

`NB_LINE`: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

`NB_LINE_INSERTED`: the number of rows inserted. This is an After variable and it returns an integer.

`ERROR_MESSAGE`: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tELTSybaseOutput is intended for use with the tELTMysqlInput and tELTSybaseMap components. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>ELT components only handle schema information. They do not handle actual data flow.</td>
</tr>
<tr>
<td>Limitation</td>
<td>This component requires installation of its related jar files.</td>
</tr>
</tbody>
</table>

### Related scenarios

- **Scenario 1: Aggregating table columns and filtering** on page 744
- **Scenario 2: ELT using an Alias table** on page 748
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTTeradataInput

Provides the Teradata table schema that will be used by the tELTTeradataMap component to generate the SQL SELECT statement.

The three ELT Teradata components are closely related, in terms of their operating conditions. These components should be used to handle Teradata database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

tELTTeradataInput Standard properties

These properties are used to configure tELTTeradataInput running in the Standard Job framework. The Standard tELTTeradataInput component belongs to the ELT family. The component in this framework is generally available.

Basic settings

| Schema and Edit schema | A schema is a row description, that is to say, it defines the nature and number of fields to be processed. The schema is either built-in or remotely stored in the Repository. The Schema defined is then passed on to the ELT Mapper to be included to the Insert SQL statement. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
- View schema: choose this option to view the schema only.  
- Change to built-in property: choose this option to change the schema to Built-in for local changes.  
- Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in</td>
<td>The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Repository</td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Default Table Name</td>
<td>Enter a default name for the table, between double quotation marks.</td>
</tr>
</tbody>
</table>
Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at component level. |

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | tELTTeradataInput is to be used along with the tELTTeradataMap. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name |

Note:
Note that the ELT components do not handle actual data flow but only schema information.

Related scenarios

- **Scenario 1: Aggregating table columns and filtering** on page 744
- **Scenario 2: ELT using an Alias table** on page 748
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTTeradataMap

Builds the SQL SELECT statement using the table schema(s) provided by one or more tELTTeradataIn
put components.

The three ELT Teradata components are closely related, in terms of their operating conditions. These
components should be used to handle Teradata database table schemas to generate SQL statements,
including clauses, which are to be executed in the database output table defined.

tELTTeradataMap Standard properties

These properties are used to configure tELTTeradataMap running in the Standard Job framework.
The Standard tELTTeradataMap component belongs to the ELT family.
The component in this framework is generally available.

Basic settings

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. | Note:
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:
1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see Talend Studio User Guide. |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ELT Teradata Map editor</td>
<td>The ELT Map editor allows you to define the output schema as well as build graphically the SQL statement to be executed. The column names of schema can be different from the column names in the database.</td>
<td></td>
</tr>
<tr>
<td>Style link</td>
<td>Select the way in which links are displayed. Auto: By default, the links between the input and output schemas and the Web service parameters are in the form of curves. Bezier curve: Links between the schema and the Web service parameters are in the form of curve.</td>
<td></td>
</tr>
</tbody>
</table>
Line: Links between the schema and the Web service parameters are in the form of straight lines. This option slightly optimizes performance.

Property type

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Either Built-in or Repository.</td>
<td></td>
</tr>
</tbody>
</table>

Built-in: No property data stored centrally.

Repository: Select the Repository file where Properties are stored. The following fields are pre-filled in using fetched data.

Host

Database server IP address

Database

Name of the database

Username and Password

DB user authentication data.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

Advanced settings

Query band

Select this check box to use the Teradata Query Banding feature to add metadata to the query to be processed, such as the user running the query. This can help you, for example, identify the origin of this query.

Once selecting the check box, the Query Band parameters table is displayed, in which you need to enter the metadata information to be added. This information takes the form of key/value pairs, for example, DpID in the Key column and Finance in the Value column.

This check box actually generates the SET QUERY_BAND FOR SESSION statement with the key/value pairs declared in the Query Band parameters table. For further information about this statement, see [http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/SQL_Reference/B035_1144_111A/End_Logging-Syntax.027.143.html](http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/SQL_Reference/B035_1144_111A/End_Logging-Syntax.027.143.html).

This check box is not available when you have selected the Using an existing connection check box. In this situation, if you need to use the Query Band feature, set it in the Advanced settings tab of the Teradata connection component to be used.

tStatCatcher Statistics

Select this check box to collect log data at the component level.

Global Variables

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Variables</td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After</td>
</tr>
</tbody>
</table>
variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

**tELTTeradataMap** is used along with a **tELTTeradataIn** put and **tELTTeradataOutput**. Note that the Output link to be used with these components must faithfully reflect the name of the tables.

Note:
The ELT components do not handle actual data flow but only schema information.

Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the Use an existing connection check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Scenario: Mapping data using a subquery

This scenario describes a Job that maps the data from two input tables **PreferredSubject** and **CourseScore** to the output table **TotalScoreOfPreferredSubject** using a subquery.
The `PreferredSubject` table contains the student’s preferred subject data. To reproduce this scenario, you can load the data to the table from a CSV file like the following. For how to load data to a Teradata table, see Scenario: Loading data into a Teradata database on page 3913.

<table>
<thead>
<tr>
<th>SeqID</th>
<th>StuName</th>
<th>Subject</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Amanda</td>
<td>art</td>
<td>Amanda prefers art.</td>
</tr>
<tr>
<td>2</td>
<td>Ford</td>
<td>science</td>
<td>Ford prefers science.</td>
</tr>
<tr>
<td>3</td>
<td>Kate</td>
<td>art</td>
<td>Kate prefers art.</td>
</tr>
</tbody>
</table>

The `CourseScore` table contains the student’s subject score data. To reproduce this scenario, you can load the data to the table from a CSV file like the following. For how to load data to a Teradata table, see Scenario: Loading data into a Teradata database on page 3913.

<table>
<thead>
<tr>
<th>SeqID</th>
<th>StuName</th>
<th>Subject</th>
<th>Course</th>
<th>Score</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Amanda</td>
<td>science</td>
<td>math</td>
<td>85</td>
<td>science score</td>
</tr>
<tr>
<td>2</td>
<td>Amanda</td>
<td>science</td>
<td>physics</td>
<td>75</td>
<td>science score</td>
</tr>
<tr>
<td>3</td>
<td>Amanda</td>
<td>science</td>
<td>chemistry</td>
<td>80</td>
<td>science score</td>
</tr>
<tr>
<td>4</td>
<td>Amanda</td>
<td>art</td>
<td>chinese</td>
<td>85</td>
<td>art score</td>
</tr>
<tr>
<td>5</td>
<td>Amanda</td>
<td>art</td>
<td>history</td>
<td>95</td>
<td>art score</td>
</tr>
<tr>
<td>6</td>
<td>Amanda</td>
<td>art</td>
<td>geography</td>
<td>80</td>
<td>art score</td>
</tr>
<tr>
<td>7</td>
<td>Ford</td>
<td>science</td>
<td>math</td>
<td>95</td>
<td>science score</td>
</tr>
<tr>
<td>8</td>
<td>Ford</td>
<td>science</td>
<td>physics</td>
<td>85</td>
<td>science score</td>
</tr>
<tr>
<td>9</td>
<td>Ford</td>
<td>science</td>
<td>chemistry</td>
<td>80</td>
<td>science score</td>
</tr>
<tr>
<td>10</td>
<td>Ford</td>
<td>art</td>
<td>chinese</td>
<td>75</td>
<td>art score</td>
</tr>
<tr>
<td>11</td>
<td>Ford</td>
<td>art</td>
<td>history</td>
<td>80</td>
<td>art score</td>
</tr>
<tr>
<td>12</td>
<td>Ford</td>
<td>art</td>
<td>geography</td>
<td>85</td>
<td>art score</td>
</tr>
<tr>
<td>13</td>
<td>Kate</td>
<td>science</td>
<td>math</td>
<td>65</td>
<td>science score</td>
</tr>
<tr>
<td>14</td>
<td>Kate</td>
<td>science</td>
<td>physics</td>
<td>75</td>
<td>science score</td>
</tr>
<tr>
<td>15</td>
<td>Kate</td>
<td>science</td>
<td>chemistry</td>
<td>80</td>
<td>science score</td>
</tr>
<tr>
<td>16</td>
<td>Kate</td>
<td>art</td>
<td>chinese</td>
<td>85</td>
<td>art score</td>
</tr>
<tr>
<td>17</td>
<td>Kate</td>
<td>art</td>
<td>history</td>
<td>80</td>
<td>art score</td>
</tr>
<tr>
<td>18</td>
<td>Kate</td>
<td>art</td>
<td>geography</td>
<td>95</td>
<td>art score</td>
</tr>
</tbody>
</table>

Before the Job execution, there is no data in the output table `TotalScoreOfPreferredSubject`:

<table>
<thead>
<tr>
<th>SeqID</th>
<th>StuName</th>
<th>PreferredSubject</th>
<th>TotalScore</th>
</tr>
</thead>
</table>

**Dropping and renaming the components**

**Procedure**

1. Create a new Job and add the following components by typing their names in the design workspace or dropping them from the Palette: two `tELTTeradataInput` components, two `tELTTeradataMap` components, and one `tELTTeradataOutput` component.
2. Rename two tELTTeradataInput components to PreferredSubject and CourseScore respectively, two tELTTeradataMap components to ELTSubqueryMap and ELTMap respectively, and the tELTTeradataOutput component to TotalScoreOfPreferredSubject.

Configuring the input components

Procedure

1. Double-click PreferredSubject to open its Basic settings view.

2. In the Default Table Name field, enter an input table name. In this example, it is PreferredSubject.
3. Click the […] button next to Edit schema to define the schema of the input table PreferredSubject in the schema editor.

   Click the [+] button to add four columns, SeqID with the DB Type set to INTEGER, StuName, Subject, and Detail with the DB Type set to VARCHAR.

   Click OK to validate these changes and close the schema editor.
4. Connect PreferredSubject to ELTMap using the Link > PreferredSubject (Table) link.
5. Double-click CourseScore to open its Basic settings view.
6. In the **Default Table Name** field, enter an input table name. In this example, it is *CourseScore*.

7. Click the [...] button next to **Edit schema** to define the schema of the input table *CourseScore* in the schema editor.

   ![CourseScore Schema](image)

   Click the [+] button to add six columns, *SeqID* and *Score* with the DB Type set to **INTEGER**, *StuName*, *Subject*, *Course* and *Detail* with the DB Type set to **VARCHAR**.

   Click **OK** to validate these changes and close the schema editor.

8. Connect *CourseScore* to *ELTSubqueryMap* using the **Link > CourseScore (Table)** link.

### Configuring the output component

**Procedure**

1. Double-click *TotalScoreOfPreferredSubject* to open its **Basic settings** view.

![TotalScoreOfPreferredSubject](image)
2. In the Default Table Name field, enter an output table name. In this example, it is `TotalScoreOfPreferredSubject`.

3. Click the [...] button next to Edit schema to define the schema of the output table in the schema editor.

![Schema of TotalScoreOfPreferredSubject](image)

Click the […] button to add four columns, `SeqID` and `TotalScore` with the DB Type set to `INTEGER`, `StuName` and `PreferredSubject` with the DB Type set to `VARCHAR`. Click OK to validate these changes and close the schema editor.

**Configuring data mapping to generate a subquery**

**Procedure**

1. Click `ELTSubqueryMap` to open its Basic settings view.

![ELTSubqueryMap(tELTTeradataMap_1)](image)

Note that you do not need to specify the Teradata database connection information in the `ELTSubqueryMap` component. The connection information will be specified in the `ELTMap` component.

2. Click the [...] button next to ELT Teradata Map Editor to open its map editor.
3. Add the input table CourseScore by clicking the [+] button in the upper left corner of the map editor and then selecting the relevant table name from the drop-down list in the pop-up dialog box.

4. Add an output table by clicking the [+] button in the upper right corner of the map editor and then entering the table name TotalScore in the corresponding field in the pop-up dialog box.

5. Drag StuName, Subject, and Score columns in the input table and then drop them to the output table.

6. Click the Add filter row button in the upper right corner of the output table and select Add an other (GROUP...) clause from the pop-up menu. Then in the Additional other clauses (GROUP/ORDER BY...) field displayed, enter the clause GROUP BY CourseScore.StuName, CourseScore.Subject. Add the aggregate function SUM for the column Score of the output table by changing the expression of this column to SUM(CourseScore.Score).

7. Click the Generated SQL Select query for “table1” output tab at the bottom of the map editor to display the corresponding generated SQL statement.

   ```sql
   SELECT CourseScore.StuName, CourseScore.Subject, SUM(CourseScore.Score)
   FROM CourseScore
   GROUP BY CourseScore.StuName, CourseScore.Subject
   ```

   This SQL query will appear as a subquery in the SQL query generated by the ELTMap component.

8. Click OK to validate these changes and close the map editor.

9. Connect ELTSubqueryMap to ELTMap using the Link > TotalScore (table1) link. Note that the link is renamed automatically to TotalScore (Table_ref) since the output table TotalScore here is a reference table.
Mapping the input and output schemas

Procedure

1. Right-click ELTMap, select **Link > “New Output” (Table)** from the contextual menu and click `TotalScoreOfPreferredSubject`. In the pop-up dialog box, click **Yes** to get the schema from the target component.

2. Click ELTMap to open its **Basic settings** view.

![ELTMap(tELTTeradataMap_2)](image)

Fill in the **Host**, **Database**, **Username**, and **Password** fields with the Teradata database connection information.

3. Click the [..] button next to **ELT Teradata Map Editor** to open its map editor.

![Talend Open Studio for Data Integration - tELTTeradataMap - tELTTeradataMap_2](image)

4. Add the input table *PreferredSubject* by clicking the [+] button in the upper left corner of the map editor and selecting the relevant table name from the drop-down list in the pop-up dialog box. Do the same to add another input table *TotalScore*.
5. Drag the StuName column in the input table PreferredSubject and drop it to the corresponding column in the input table TotalScore. Then select the Explicit join check box for the StuName column in the input table TotalScore.
   Do the same for the Subject column.

6. Drag the SeqID column in the input table PreferredSubject and drop it to the corresponding column in the output table.
   Do the same to drag the StuName and Subject columns in the input table PreferredSubject and the Score column in the input table TotalScore and drop them to the corresponding column in the output table.

7. Click the Generated SQL Select query for “table2” output tab at the bottom of the map editor to display the corresponding generated SQL statement.

The SQL query generated in the ELTSUBQUERY component appears as a subquery in the SQL query generated by this component. Alias will be automatically added for the selected columns in the subquery.

8. Click OK to validate these changes and close the map editor.

**Executing the Job**

**Procedure**

1. Press Ctrl + S to save the Job.
2. Press F6 to run the Job.
The select statement is generated and the mapping data are written into the output table.

**Related scenarios**

- Scenario 1: Aggregating table columns and filtering on page 744
- Scenario 2: ELT using an Alias table on page 748
- Aggregating Snowflake data using context variables as table and connection names on page 724
tELTTeradataOutput

Performs the action (insert, update or delete) on data in the specified Teradata table through the SQL statement generated by the tELTTeradataMap component.

The three ELT Teradata components are closely related, in terms of their operating conditions. These components should be used to handle Teradata database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

**tELTTeradataOutput Standard properties**

These properties are used to configure tELTTeradataOutput running in the Standard Job framework. The Standard tELTTeradataOutput component belongs to the ELT family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Action on data</th>
<th>On the data of the table defined, you can perform the following operation:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, Job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Updates entries in the table.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Deletes the entries which correspond to the entry flow.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
</tbody>
</table>

| Built-in: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide. |
**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

**Where clauses for (for UPDATE and DELETE only)** Enter a clause to filter the data to be updated or deleted during the update or delete operations.

**Default Table Name** Enter a default name for the table, between double quotation marks.

Note that the table must exist already. If it does not exist, you can use `tCreateTable` to create one first. For more information about `tCreateTable`, see `tCreateTable` on page 541.

**Table name from connection name is variable** Select this check box when the name of the connection to this component is set to a variable, such as a context variable.

**Use different table name** Select this check box to enter a different output table name, between double quotation marks, in the **Table name** field which appears.

**Mapping** Specify the metadata mapping file for the database to be used. The metadata mapping file is used for the data type conversion between database and Java. For more information about the metadata mapping, see the related documentation for Type mapping.

**Advanced settings**

**tStatCatcher Statistics** Select this check box to gather the Job processing metadata at a Job level as well as at component level.

**Global Variables**

**Global Variables**

- **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

- **NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.
### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tELTTeradataOutput is to be used along with the tELTTeradataMap. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note:</td>
</tr>
<tr>
<td></td>
<td>Note that the ELT components do not handle actual data flow but only schema information.</td>
</tr>
</tbody>
</table>

### Limitation

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |

### Related scenarios

- **Scenario 1:** Aggregating table columns and filtering on page 744
- **Scenario 2:** ELT using an Alias table on page 748
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTVerticaInput

Provides the Vertica table schema that will be used by the tELTVerticaMap component to generate the SQL SELECT statement.

The three ELT Vertica components are closely related, in terms of their operating conditions. These components should be used to handle Vertica database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

tELTVerticaInput Standard properties

These properties are used to configure tELTVerticaInput running in the Standard Job framework.
The Standard tELTVerticaInput component belongs to the ELT family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
<td></td>
</tr>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td>Default Table Name</td>
<td>Type in the default table name.</td>
</tr>
<tr>
<td>Default Schema Name</td>
<td>Type in the default schema name.</td>
</tr>
</tbody>
</table>
**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE:</strong> the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td>For further information about variables, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th><strong>Usage rule</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tELTVerticalInput</strong> is used along with <strong>tELTVerticaMap</strong>. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td>The ELT components do not handle actual data flow but only schema information.</td>
</tr>
</tbody>
</table>

**Related scenarios**

- **Scenario 1:** Aggregating table columns and filtering on page 744
- **Scenario 2:** ELT using an Alias table on page 748
- **Scenario:** Mapping data using a subquery on page 800
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tELTVerticaMap

Builds the SQL SELECT statement using the table schema(s) provided by one or more tELTVerticalInput components.

The three ELT Vertica components are closely related, in terms of their operating conditions. These components should be used to handle Vertica database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

**tELTVerticaMap Standard properties**

These properties are used to configure tELTVerticaMap running in the Standard Job framework.

The Standard tELTVerticaMap component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>DB Version</th>
<th>Select the version of the Vertica database being used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>ELT Vertica Map Editor</th>
<th>The ELT Map editor allows you to define the output schema as well as build graphically the SQL statement to be executed. The column names of the schema can be different from the column names in the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Style link</td>
<td>Select a way in which links are displayed.</td>
</tr>
</tbody>
</table>

- **Auto:** By default, the links between the input and output schemas and the Web service parameters are in the form of curves.
- **Bezier curve**: The links between the schema and the Web service parameters are in the form of curve.
- **Line (fastest)**: The links between the schema and the Web service parameters are in the form of straight lines. This option slightly optimizes performance.

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-In</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

| Host            | Type in the IP address or hostname of the database. |
| Port            | Type in the listening port number of the database. |
| Database        | Type in the name of the database you want to use. |
| Additional JDBC Parameters | Specify additional connection properties for the database connection you are creating. |
| Username and Password | Type in the database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |

**Advanced settings**

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tELTVerticaMap is used along with tELTVerticaInput and tELTVerticaOutput. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note:</td>
</tr>
<tr>
<td></td>
<td>The ELT components do not handle actual data flow but only schema information.</td>
</tr>
</tbody>
</table>

Related scenarios

- Scenario 1: Aggregating table columns and filtering on page 744
- Scenario 2: ELT using an Alias table on page 748
- Scenario: Mapping data using a subquery on page 800, a related scenario using subquery
- Aggregating Snowflake data using context variables as table and connection names on page 724
**tELTVerticaOutput**

Performs the action (insert, update or delete) on data in the specified Vertica table through the SQL statement generated by the tELTVerticaMap component.

The three ELT Vertica components are closely related, in terms of their operating conditions. These components should be used to handle Vertica database table schemas to generate SQL statements, including clauses, which are to be executed in the database output table defined.

**tELTVerticaOutput Standard properties**

These properties are used to configure tELTVerticaOutput running in the Standard Job framework. The Standard tELTVerticaOutput component belongs to the ELT family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Action on data</th>
<th>On the data of the table defined, you can perform one of the following operations:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Insert</strong>: Add new entries to the table. If duplicates are found, Job stops.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update</strong>: Updates entries in the table.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Delete</strong>: Deletes entries which correspond to the entry flow.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
</tbody>
</table>

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No**.
upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th><strong>Sync columns</strong></th>
<th>Click this button to retrieve the schema from the previous component connected in the Job.</th>
</tr>
</thead>
</table>
| **Where clauses (for UPDATE and DELETE only)** | Enter a clause to filter the data to be updated or deleted during the update or delete operation.  
This field is available only when Update or Delete is selected from the Action on data drop-down list. |
| **Default Table Name** | Type in the default table name. |
| **Default Schema Name** | Type in the default schema name. |
| **Table name from connection name is variable** | Select this check box when the name of the connection to this component is set to a variable, such as a context variable. |
| **Use different table name** | Select this check box to use a different output table name. |
| **Table name** | Type in the output table name.  
This field is available only when the Use different table name check box is selected. |

**Advanced settings**

| **Direct** | Select this check box to write the data directly to disk, bypassing memory.  
This check box is not visible when the Set SQL Label check box is selected. |
| **Set SQL Label** | Select this check box and specify the label that identifies the query. For more information, see How to label queries for profiling.  
This check box is not visible when the Direct check box is selected. |
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| **Global Variables** |  
|----------------------|------------------------------------------------------------------------------------------------|
| **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
**NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tELTVerticaOutput is used along with the tELTVerticaMap. Note that the Output link to be used with these components must correspond strictly to the syntax of the table name.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note: The ELT components do not handle actual data flow but only schema information.</td>
</tr>
</tbody>
</table>

## Related scenarios

- **Scenario 1: Aggregating table columns and filtering** on page 744
- **Scenario 2: ELT using an Alias table** on page 748
- **Scenario: Mapping data using a subquery** on page 800
- **Aggregating Snowflake data using context variables as table and connection names** on page 724
tESBConsumer

Calls the defined method from the invoked Web service and returns the class as defined, based on the given parameters.

**tESBConsumer Standard properties**

These properties are used to configure tESBConsumer running in the Standard Job framework.

The Standard tESBConsumer component belongs to the ESB family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Service configuration</th>
<th>Description of Web service bindings and configuration. The <strong>Endpoint</strong> field gets filled in automatically upon completion of the service configuration.</th>
</tr>
</thead>
</table>

**Input Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see **Talend Studio User Guide**.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see **Talend Studio User Guide**.

**Response Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:
- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in:** The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository:** The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

**Fault Schema** and **Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in:** The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository:** The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

**Use Service Registry**

This option is only available if you subscribed to Talend Enterprise ESB solutions.

Select this check box to enable the Service Registry. It provides dynamic endpoint lookup and allows services to be redirected based upon information retrieved from the registry. It works in runtime only.

Enter the authentication credentials in the **Username** and **Password** field.
If SAML token is registered in the service registry, you need to specify the client’s role in the Role field. You can also select the Propagate Credentials check box to make the call on behalf of an already authenticated user by propagating the existing credentials. You can enter the username and the password to authenticate via STS to propagate using username and password, or provide the alias, username and the password to propagate using certificate. For more information, see the Use Authentication option. Select the Encryption/Signature body check box to enable XML Encryption/XML Signature. For more information, see the chapter about XKMS Service in the Talend ESB Infrastructure Services Configuration Guide.

In the Correlation Value field, specify a correlation ID or leave this field empty. For more information, see the Use Business Correlation option.

For more information about how to set up and use the Service Registry, see the Talend Administration Center User Guide and Talend ESB Infrastructure Services Configuration Guide.

<table>
<thead>
<tr>
<th>Use Service Locator</th>
<th>Maintains the availability of the service to help meet demands and service level agreements (SLAs). This option will not show if the Use Service Registry check box is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Service Activity Monitor</td>
<td>Captures events and stores this information to facilitate in-depth analysis of service activity and track-and-trace of messages throughout a business transaction. This can be used to analyze service response times, identify traffic patterns, perform root cause analysis and more. This option is disabled when the Use Service Registry check box is selected if you subscribed to Talend Enterprise ESB solutions.</td>
</tr>
<tr>
<td>Use Authentication</td>
<td>Select this check box to enable the authentication option. Select from Basic HTTP, HTTP Digest, Username Token, and SAML Token (ESB runtime only). Enter a username and a password in the corresponding fields as required. Authentication with Basic HTTP, HTTP Digest, and Username Token work in both the studio and runtime. Authentication with the SAML Token works in runtime only. When SAML Token (ESB runtime only) is selected, you can either provide the user credentials to send the request or make the call on behalf of an already authenticated user by propagating the existing credentials. Select from: - Propagate using U/P: Enter the user name and the password used to authenticate against STS. - Propagate using Certificate: Enter the alias and the password used to authenticate against STS.</td>
</tr>
</tbody>
</table>

Use Service Locator

Use Service Activity Monitor

Use Authentication
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

This option will not show if the **Use Service Registry** check box is selected.

| **Use Business Correlation** | Select this check box to create a correlation ID in this component.
You can specify a correlation ID in the **Correlation Value** field. In this case the correlation ID will be passed on to the service it calls so that chained service calls will be grouped under this correlation ID. If you leave this field empty, this value will be generated automatically at runtime.
When this option is enabled, **tESBConsumer** will also extract the correlation ID from the response header and store it in the component variable for further use in the flow.
This option will be enabled automatically when the **Use Service Registry** check box is selected. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use GZip Compress</strong></td>
<td>Select this check box to compress the incoming messages into GZip format before sending.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>Select this check box to kill the Job when an error occurs.</td>
</tr>
</tbody>
</table>

### Advanced settings

| **Log messages** | Select this check box to log the message exchange between the service provider and the consumer. |
| **Service Locator Custom Properties** | This table appears when **Use Service Locator** is selected. You can add as many lines as needed in the table to customize the relevant properties. Enter the name and the value of each property between double quotation marks in the **Property Name** field and the **Property Value** field respectively. |
| **Service Activity Custom Properties** | This table appears when **Use Service Activity Monitor** is selected. You can add as many lines as needed in the table to customize the relevant properties. Enter the name and the value of each property between double quotation marks in the **Property Name** field and the **Property Value** field respectively. |
| **Connection time out(second)** | Set a value in seconds for Web service connection time out.
This option only works in the studio. To use it after the component is deployed in runtime:
1. Create a configuration file with the name `org.apache.cxf.http.conduits-<endpoint_name>.cfg` in the `<TalendRuntime Path>/container/etc/` folder. |
2. Specify the url of the Web service and the client.ConnectionTimeout parameter in milliseconds in the configuration file. If you need to use the Receive time out option, specify the client.ReceiveTimeout in milliseconds too. The url can be a full endpoint address or a regular expression containing wild cards, for example:

```
url = http://localhost:8040/*
client.ConnectionTimeout=1000000
client.ReceiveTimeout=20000000
```


<table>
<thead>
<tr>
<th>Receive time out(second)</th>
<th>Set a value in seconds for server answer. This option only works in the studio. For how to use it after the component is deployed in runtime, see the Connection time out option.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable Chunking</td>
<td>Select this check box to disable encoding the payload as chunks. In general, chunking will perform better as the streaming can take place directly. But sometimes the payload is truncated with chunking enabled. If you are getting strange errors when trying to interact with a service, try turning off chunking to see if that helps.</td>
</tr>
</tbody>
</table>
| Trust server with SSL/TrustStore file and TrustStore password | Select this check box to validate the server certificate to the client via an SSL protocol and fill in the corresponding fields:  

TrustStore file: Enter the path (including filename) to the certificate TrustStore file that contains the list of certificates that the client trusts.  

TrustStore password: Enter the password used to check the integrity of the TrustStore data. |
| Use http proxy/Proxy host, Proxy port, Proxy user, and Proxy password | Select this check box if you are using a proxy server and fill in the necessary information.  

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| HTTP Headers | Click [+] as many times as required to add the name-value pair(s) for HTTP headers to define the parameters of the requested HTTP operation. |
| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

| Global Variables | NB_LINE: the number of rows processed. This is an After variable and it returns an integer. |
CORRELATION_ID: the correlation ID by which chained service calls will be grouped. This is a Flow variable and it returns a string.

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

HTTP_RESPONSE_CODE: HTTP response status code. This is an After variable and it returns an Integer.

HTTP_HEADERS: the set of HTTP headers from response. This is a Flow variable and it returns map object java.util.Map<String, java.util.List<?>>. Header name is represented by map key. Header values are represented by java.util.List<?>. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

This component can be used as an intermediate component. It requires to be linked to an output component.

Dynamic settings

Click the [+ ] button to add a row in the table and fill the Code field with a context variable to turn on or off the Use Authentication or Use HTTP proxy option dynamically at runtime. You can add two rows in the table to set both options.

Once a dynamic parameter is defined, the corresponding option becomes highlighted and unusable in the Basic settings view or Advanced settings view.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Limitation

A JDK is required for this component to operate.

Scenario 1: Using tESBConsumer to retrieve the valid email

This scenario describes a Job that uses a tESBConsumer component to retrieve the valid email.
Dropping and linking the components

Procedure

1. Drop the following components from the Palette onto the design workspace: a tFixedFlowInput, a tESBConsumer, two tXMLMap, and two tLogRow components.
2. Right-click the tFixedFlowInput component, select Row > Main from the contextual menu and click the first tXMLMap component.
3. Right-click the tXMLMap component, select Row > *New Output* (Main) from the contextual menu and click the tESBConsumer component. Enter payload in the popup dialog box to name this row and accept the propagation that prompts you to get the schema from the tESBConsumer component.
4. Right-click the tESBConsumer component, select Row > Response from the contextual menu and click the second tXMLMap component.
5. Right-click the second tXMLMap component, select Row > *New Output* (Main) from the contextual menu and click the second tLogRow component. Enter response in the popup dialog box to name this row.
6. Right-click the tESBConsumer component again, select Row > Fault from the contextual menu and click the other tLogRow component.

Configuring the components

The tLogRow components will monitor the exchanges from the response and fault messages and does not need any configuration. Press Ctrl+S to save your job.

Configuring the tESBConsumer component

About this task

In this scenario, a public web service which is available at http://www.webservicex.net/ValidateEmail.asmx will be called by the tESBConsumer component to returns true or false for an email address. You can view the WSDL definition of the service at http://www.webservicex.net/ValidateEmail.asmx?WSDL for the service description.

Procedure

1. In the design workspace, double-click the tESBConsumer component to open its Basic settings view in the Component tab.
2. Click the three-dot button next to **Service configuration**.

3. In the dialog box that appears, type in: `http://www.webservicex.net/ValidateEmail.asmx?WSDL` in the **WSDL** field and click the refresh button to retrieve port name and operation name. In the **Port Name** list, select the port you want to use, **ValidateEmailSoap** in this example.

Select the **Populate schema to repository on finish** to retrieve the schema from the WSDL definition, which will be used by the `tFixedFlowInput` component. This option is only available to users of **Talend Studio** with ESB. If you don’t have this option, please ignore it. The schema can be created manually in the `tFixedFlowInput` component, which will be shown later.

Click **Finish** to validate your settings and close the dialog box.
4. Click the **Advanced settings** view in the **Component** tab.

5. Select the **Log messages** check box to show the exchange log in the execution console.

**Configuring the tFixedFlowInput component**

**Procedure**

1. Double-click the tFixedFlowInput component to open its **Basic settings** view in the **Component** tab.

2. For users of **Talend Studio** with ESB who have retrieved the schema from the service WSDL definition in the configuration of the **tESBConsumer** component, select **Repository** from the **Schema** list. Then click the [...] of the next field to show the **Repository Content** dialog box. Select the metadata under the **IsValidEmail** node to use it as the schema of the input message. Click **OK** to close the dialog box.

For users of **Talend Studio** without ESB, please go to the next step.
3. For users of *Talend Studio* without ESB, the schema need to be created manually. Select **Built-In** from the **Schema** list.

Click the three-dot button next to **Edit Schema**. In the schema dialog box, click the plus button to add a new line of **String** type and name it **Email**. Click **OK** to close the dialog box.
4. In the **Number of rows** field, set the number of rows as 1.

5. In the **Mode** area, select **Use Single Table** and input the following request in double quotation marks into the **Value** field:

   nomatter@gmail.com

### Configuring the tXMLMap component in the input flow

**About this task**

Talend data integration uses schemas based on rows and columns since it has roots in relational data warehouse integration. But SOAP messages uses the XML format. XML is hierarchical and supports richer structure than rows or columns. So we need the tXMLMap to convert from the relational row/column structure to the schema expected by the SOAP service.

**Procedure**

1. In the design workspace, double-click the tXMLMap component to open the **Map Editor**.

2. In the output table, right-click the root node and select **Rename** from the contextual menu. Enter `IsValidEmail` in the dialog box that appears.

3. Right-click the `IsValidEmail` node and select **Set A Namespace** from the contextual menu. Enter `http://www.webservicex.net` in the dialog box that appears.

4. Right-click the `IsValidEmail` node again and select **Create Sub-Element** from the contextual menu. Enter `Email` in the dialog box that appears.

5. Right-click the `Email` node and select **As loop element** from the contextual menu.

6. Click the `Email` node in the input table and drop it to the **Expression** column in the row of the `Email` node in the output table.
7. Click OK to validate the mapping and close the Map Editor.

**Configuring the tXMLMap component in the output flow**

**About this task**

The tXMLMap in the output flow will convert the response message from the XML format to the row/column structure.

**Procedure**

1. In the design workspace, double-click the tXMLMap component in the output flow to open the Map Editor.
2. In the input table, right-click the root node and select Rename from the contextual menu. Enter IsValidEmailResponse in the dialog box that appears.
3. Right-click the IsValidEmailResponse node and select Set A Namespace from the contextual menu. Enter http://www.webservicex.net in the dialog box that appears.
4. Right-click the IsValidEmailResponse node again and select Create Sub-Element from the contextual menu. Enter IsValidEmailResult in the dialog box that appears.
5. Right-click the IsValidEmailResult node and select As loop element from the contextual menu.
6. On the lower right part of the map editor, click [+] to add a row of String type to the output table and name it response.
7. Click the IsValidEmailResult node in the input table and drop it to the Expression column in the row of the response node in the output table.
8. Click OK to validate the mapping and close the Map Editor.

**Executing the Job**

Click the Run view to display it and click the Run button to launch the execution of your Job. You can also press F6 to execute it. In the execution log you will see:
The email address `nomatter@gmail.com` is returned as false. The input and output SOAP messages in XML are also shown in the console.

**Scenario 2: Using tESBConsumer with custom SOAP Headers**

This scenario is similar to the previous one. It describes a Job that uses a `tESBConsumer` component to retrieve a valid email address with custom SOAP headers in the request message.
Dropping and linking the components

Procedure
1. Drop the following components from the Palette onto the design workspace: a tESBConsumer, a tMap, two tFixedFlowInput, three tXMLMap, and two tLogRow.
2. Connect each of the tFixedFlowInput with a tXMLMap using the Row > Main connection.
3. Right-click the first tXMLMap, select Row > *New Output* (Main) from the contextual menu and click tMap. Enter payload in the popup dialog box to name this row. Repeat this operation to connect another tXMLMap to tMap and name the output row header.
4. Right-click the tMap component, select Row > *New Output* (Main) from the contextual menu and click the tESBConsumer component. Enter request in the popup dialog box to name this row and accept the propagation that prompts you to get the schema from the tESBConsumer component.
5. Right-click the tESBConsumer component, select Row > Response from the contextual menu and click the third tXMLMap component.
6. Right-click the third tXMLMap component, select Row > *New Output* (Main) from the contextual menu and click one of the tLogRow components. Enter response in the popup dialog box to name this row.
7. Right-click the tESBConsumer component again, select Row > Fault from the contextual menu and click the other tLogRow component.

Configuring the components

The tLogRow components will monitor the exchanges from the response and fault messages and does not need any configuration. Press Ctrl+S to save your Job.

Configuring the tESBConsumer component

About this task

In this scenario, a public web service which is available at http://www.webservicex.net/ValidateEmail.asmx will be called by the tESBConsumer component to returns true or false for an email address. You can view the WSDL definition of the service at http://www.webservicex.net/ValidateEmail.asmx?WSDL for the service description.
Procedure

1. In the design workspace, double-click the tESBConsumer component to open its Basic settings view in the Component tab.

![Image of tESBConsumer component settings]

2. Click the [...] button next to Service configuration.

![Image of Configure component with Web Service operation]

3. In the dialog box that appears, type in: http://www.webservicex.net/ValidateEmail.asmx?WSDL in the WSDL field and click the refresh button to retrieve port name and operation name. In the Port Name list, select the port you want to use, ValidateEmailSoap in this example. Click OK to validate your settings and close the dialog box.

Select the Populate schema to repository on finish to retrieve the schema from the WSDL definition, which will be used by the tFixedFlowInput component. This option is only available to users of Talend Studio with ESB. If you don’t have this option, please ignore it. The schema can be created manually in the tFixedFlowInput component, which will be shown later.
Click Finish to validate your settings and close the dialog box.

4. In the Advanced settings view, select the Log messages check box to log the content of the messages.

Configuring the tFixedFlowInput components

Procedure

1. Double-click the first tFixedFlowInput component to open its Basic settings view in the Component tab.

2. For users of Talend Studio with ESB who have retrieved the schema from the service WSDL definition in the configuration of the tESBConsumer component, select Repository from the Schema list. Then click the [...] of the next field to show the Repository Content dialog box. Select the metadata under the IsIsValidEmail node to use it as the schema of the input message. Click OK to close the dialog box.
   
   For users of Talend Studio without ESB, please go to the next step.
3. For users of *Talend Studio* without ESB, the schema needs to be created manually. Select **Built-In** from the **Schema** list.

Click the [...] button next to **Edit Schema**. In the schema dialog box, click the [+•] button to add a new line of **String** type and name it **Email**. Click **OK** to close the dialog box.
4. In the **Number of rows** field, set the number of rows as 1.

5. In the **Mode** area, select **Use Single Table** and enter "nomatter@gmail.com" into the **Value** field, which is the payload of the request message.

6. Configure the second **tFixedFlowInput** as the first one, except for its schema. Add two rows of **String** type to the schema and name them *id* and *company* respectively.

Give the value *Hello world!* to *id* and *Talend* to *company*, which are the headers of the request message.
Configuring the tXMLMap components in the input flow

About this task

Talend data integration uses schemas based on rows and columns since it has roots in relational data warehouse integration. But SOAP messages use the XML format. XML is hierarchical and supports richer structure than rows or columns. So we need the tXMLMap to convert from the relational row/column structure to the schema expected by the SOAP service.

Procedure

1. In the design workspace, double-click the first tXMLMap component to open the Map Editor.
2. In the output table, right-click the root node and select Rename from the contextual menu. Enter IsValidEmail in the dialog box that appears.
3. Right-click the IsValidEmail node and select Set A Namespace from the contextual menu. Enter http://www.webservicex.net in the dialog box that appears.
4. Right-click the IsValidEmail node again and select Create Sub-Element from the contextual menu. Enter Email in the dialog box that appears.
5. Right-click the Email node and select As loop element from the contextual menu.
6. Click the Email node in the input table and drop it to the Expression column in the row of the Email node in the output table.
7. Click **OK** to validate the mapping and close the **Map Editor**.

8. Configure the other **tXMLMap** in the same way. Add a row of **Document** type to the output table and name it **header**. Create two sub-elements to it, **id** and **company**. Map the **id** and the **company** nodes in the input table to the corresponding nodes in the output table.
Configuring the tMap component

Procedure

1. In the design workspace, double-click tMap to open the Map Editor.

2. On the lower right part of the map editor, click \([+]\) to add two rows of Document type to the output table and name them payload and headers respectively.

3. Click the payload node in the input table and drop it to the Expression column in the row of the payload node in the output table.

4. Click the header node in the input table and drop it to the Expression column in the row of the headers node in the output table.

Configuring the tXMLMap component in the output flow

About this task

The tXMLMap in the output flow will convert the response message from the XML format to the row/column structure.

Procedure

1. In the design workspace, double-click the tXMLMap component in the output flow to open the Map Editor.

2. In the input table, right-click the root node and select Rename from the contextual menu. Enter IsValidEmailResponse in the dialog box that appears.

3. Right-click the IsValidEmailResponse node and select Set A Namespace from the contextual menu. Enter http://www.webservicex.net in the dialog box that appears.
4. Right-click the `IsValidEmailResponse` node again and select **Create Sub-Element** from the contextual menu. Enter `IsValidEmailResult` in the dialog box that appears.

5. Right-click the `IsValidEmailResult` node and select **As loop element** from the contextual menu.

6. On the lower right part of the map editor, click `[+]` to add a row of `String` type to the output table and name it `response`.

7. Click the `IsValidEmailResult` node in the input table and drop it to the **Expression** column in the row of the `response` node in the output table.

8. Click **OK** to validate the mapping and close the **Map Editor**.

**Executing the Job**

Click the **Run** view to display it and click the **Run** button to launch the execution of your Job. You can also press **F6** to execute it.

---

842
As shown in the execution log, the email address nomatter@gmail.com is returned as false. The input and output SOAP messages in XML is also shown in the console. The SOAP header is sent with the request to the service.
tESBProviderFault

Serves a Talend Job cycle result as a Fault message of the Web service in case of a request response communication style.

It acts as Fault message of the Web Service response at the end of a Talend Job cycle.

**tESBProviderFault Standard properties**

These properties are used to configure tESBProviderFault running in the Standard Job framework. The Standard tESBProviderFault component belongs to the ESB family.

*This component is relevant only when used with one of the Talend solutions with ESB, as it should be used with the Service Repository node and the Data Service creation related wizard(s).*

**Basic settings**

| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
• **View schema**: choose this option to view the schema only.  
• **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. Click **Sync columns** to retrieve the schema from the previous component connected in the Job. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>

**EBS service settings**

| Fault title | Value of the faultString column in the Fault message.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td></td>
</tr>
</tbody>
</table>
The Row > Fault flow of tESBConsumer has a pre-defined schema whose column, faultString,
is filled up with the content of the field Fault title of tESBProviderFault.

Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

Global Variables

| Global Variables | NB_LINE: the number of rows processed. This is an After variable and it returns an integer. ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component should only be used with the tESBProviderRequest component. |
| Limitation | A JDK is required for this component to operate. |

Scenario: Requesting airport names based on country codes

This scenario applies only to a Talend solution with ESB.

This scenario involves two Jobs, one as the data service provider and the other as data service consumer. The former listens to the requests from the consumer via tESBProviderRequest, matches the country code wrapped in the request against a MySQL database table that has the country code/airport pairs via tMap, and finally returns the correct airport name via tESBProviderResponse or if no matches are found, the error message via tESBProviderFault. The Consumer sends requests to the Provider and receives the airport information or error reminders via tESBConsumer.

Building the data service provider to publish a service

The data service airport has already been defined under the Services node of the Repository tree view. Its schema has three major elements as shown below:
Assigning a Job to the defined service

Procedure

1. Right-click `getAirportInformationByISOCountryCode` under the Web service `airport` and from the contextual menu, select **Assign Job**.

2. In the [Operation Choice] window, select **Create a new Job and Assign it to this Service Operation**.

3. Click **Next** to open the Job description window. The Job name `airportSoap_getAirportInformationByISOCountryCode` is automatically filled in.
4. Click **Finish** to create the Job and open it in the workspace. Three components are already available.

---

**Adding components to arrange the data flow**

**Procedure**

1. Drop **tXMLMap** and **tMysqlInput** from the **Palette** to the workspace.
2. Link **tESBProviderRequest** to **tXMLMap** using a **Row > Main** connection.
3. Link **tMysqlInput** to **tXMLMap** using a **Row > Main** connection.
4. Link **tXMLMap** to **tESBProviderResponse** using a **Row > *New Output* (Main)** connection.
   - In the **new Output name** pop-up window, enter the output table name, `airport_response`.
   - Click **OK** in the pop-up window that asks whether to get the schema of the target component.
5. Link tXMLMap to tESBProviderFault using a **Row > New Output** (Main) connection. In the **new Output name** pop-up window, enter the output table name, *fault_message*. Click **OK** in the pop-up window that asks whether to get the schema of the target component.

**Configuring how requests are processed**

**Procedure**

1. Double-click **tMysqlInput** to display its **Basic settings** view.

2. Fill up the basic settings for the Mysql connection and database table. Click the [...] button to open the schema editor.
3. Click the [*] button to add two columns, id and name, with the type of string. 
Click OK to close the editor. 
Click **Guess Query** to retrieve the SQL query. 

4. Double-click **tXMLMap** to open its mapper. 

5. In the **main : row1** table of the input flow side (left), right-click the column name **payload** and from the contextual menu, select **Import from Repository**. Then the **[Metadata]** wizard is opened.
Select the schema of the request message and click OK to validate this selection. In this example, the schema is `getAirportInformationByISOCountryCode`.

6. Do the same to import the hierarchical schemas for the response/fault messages (right). In this example, these schemas are `getAirportInformationByISOCountryCodeResponse` and `getAirportInformationByISOCountryCodeFault` respectively.

7. Then to create the join to the lookup data, drop the `CountryAbbrviation` node from the main flow onto the `id` column of the lookup flow.

8. On the lookup flow table, click the wrench icon on the upper right corner to open the setting panel.
   Set `Lookup Model` as `Reload at each row`, `Match Model` as `All matches` and `Join Model` as `Inner join`.

9. On the `airport_response` output flow table, click the wrench icon on the upper right corner to open the setting panel.
   Set the `All in one` option as `true`. This ensures that only one response is returned for each request if multiple airport matches are found in the database.

10. On the `fault_message` output flow table, click the wrench icon on the upper right corner to open the setting panel.
    Set the `Catch Lookup Inner Join Reject` option as `true` to monitor the mismatches between the country code in the request and the records in the database table. Once such a situation occurs, a fault message will be generated by `tESBConsumer` and outputted via its `Row > Fault` flow.

Note:
The Row > Fault flow of tESBConsumer has a predefined schema in which the faultString column is filled with the content of the field Fault title of tESBProviderFault.

11. Drop the name column in the lookup flow onto the Expression area next to the tns:getAirportInformationByISOCountryCodeResult node in the airport_response output flow. Drop the tns:CountryAbbreviation node in the main flow onto the Expression area next to the tns:getAirportInformationByISOCountryCodeFaultString node in the fault_message output flow. This way, the incorrect country code in the request will be shown in the faultDetail column of the Row > Fault flow of tESBConsumer.

Click OK to close the editor and validate this configuration.

12. Double-click tESBProviderFault to display its Basic settings view:

13. In the field Fault title, enter the context variable context.fault_message.

For how to define context variables, see Talend Studio User Guide.

Publishing the service to listen to requests

Procedure
1. Press Ctrl +S to save the Job.
2. Press F6 to run this Job.

Results
The data service is published and will listen to all the requests until you click the Kill button to stop it as by default, the Keep listening option of tESBProviderRequest is selected automatically.

Now is the time to configure the consumer Job that interacts with the data service.
Building the data service consumer to request the service

Built upon **tESBConsumer**, the consumer Job sends two requests that contain the country codes to the Web service for the relevant airport names. If wrong country code is wrapped in the request, the error message will be returned. The country codes and the MySQL database records are as follows:

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN</td>
<td>Capital Airport</td>
</tr>
<tr>
<td>DE</td>
<td>Frankfurt Airport</td>
</tr>
<tr>
<td>FR</td>
<td>Roissy Airport</td>
</tr>
<tr>
<td>*</td>
<td>(NULL)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>country_code_tbd</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CC</td>
</tr>
<tr>
<td>2</td>
<td>CN</td>
</tr>
</tbody>
</table>

Dropping and linking the components

**Procedure**

1. Drop a **tFileInputDelimited**, a **tXMLMap**, a **tESBConsumer** and two **tLogRow** from the **Palette** to the workspace.
2. Rename one **LogRow** as **response** and the other as **fault_message**.
3. Link **tFileInputDelimited** to **tXMLMap** using a **Row > Main** connection.
4. Link **tXMLMap** to **tESBConsumer** using a **Row > *New Output*(Main)** connection.
   In the **new Output name** pop-up window, enter the output table name, for example **request**.
   Click **OK** in the pop-up window that asks whether to get the schema of the target component.
5. Link **tESBConsumer** to **response** using the **Row > Response** connection.
6. Link **tESBConsumer** to **fault_message** using the **Row > Fault** connection.

Configuring the components

**Procedure**

1. Double-click **tFileInputDelimited** to open its **Basic settings** view.
2. In the **File name/stream** field, enter the context variable for the file that has the country codes, `context.filepath`.

3. Click the `[...]` button to open the schema editor.

4. Click the `[+]` button to add a column, `country_code`, for example, with the type of string. Click **OK** to close the editor.

5. Double-click **tXMLMap** to open its Map editor.
6. In the request table of the output flow side, right-click the column name payload and from the contextual menu, select **Import from Repository**. Then the [Metadata] wizard is opened.

Select the schema of the request message and click **OK** to validate this selection. In this example, the schema is `getAirportInformationByISOCountryCode`. 
7. Drop the *country_code* column in the main flow onto the **Expression** area next to the *tns:CountryAbb reviation* node in the *request* output flow. Click **OK** to close the editor and validate this configuration.

8. Double-click **tESBConsumer** to open its service configuration wizard:

![Configure component with Web Service operation](image)

9. Click the **Browse...** button to select the desired WSDL file. The **Port name** and **Operation** are automatically filled up once the WSDL file is selected. Click **OK** to close the wizard.

10. Double-click **response** to open its **Basic settings** view:

![response(tLogRow_1)](image)

11. Select **Vertical (each row is a key/value list)** and then **Print label** for a better view of the results. Do the same to the other **tLogRow, fault_message**.

**Executing the Job**

**Procedure**

1. Press **Ctrl +S** to save the Job.
2. Press **F6** to run this Job.
As shown above, two messages are returned, one giving the airport name that matches the country code CN and the other giving the error details caused by the country code CC.
tESBProviderRequest

Wraps Talend Job as web service.

It waits for a request message from a consumer and passes it to the next component.

**tESBProviderRequest Standard properties**

These properties are used to configure tESBProviderRequest running in the Standard Job framework.

The Standard tESBProviderRequest component belongs to the ESB family.

*This component is relevant only when used with one of the Talend solutions with ESB, as it should be used with the Service Repository node and the Data Service creation related wizard(s).*

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
</tr>
<tr>
<td><strong>Repository</strong></td>
</tr>
</tbody>
</table>

**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

- **Built-in**: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.
- **Repository**: The schema is created and stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.

**Keep listening**

Check this box when you want to ensure that the provider (and therefore Talend Job) will continue
listening for requests after processing the first incoming request.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Log messages (Studio only)</strong></td>
<td>Select this check box to log the message exchange between the service provider and the consumer. This option works in the Studio only.</td>
</tr>
<tr>
<td><strong>Response timeout, sec</strong></td>
<td>Specify the time limit in seconds for sending response to the consumer. This parameter is necessary to avoid locking of message exchanges.</td>
</tr>
<tr>
<td><strong>Request processing queue size</strong></td>
<td>Specify the maximum number of received requests that can be processed in parallel by the components between <code>tESBProviderRequest</code> and <code>tESBProviderResponse</code>. Note that this parameter is different from the <code>queueSize</code> in the <code>&lt;TalendRuntimePath&gt;/etc/\org.apache.cxf.workqueues-default.cfg</code> which defines pool configuration for incoming requests on CXF level.</td>
</tr>
<tr>
<td><strong>Request processing timeout, sec</strong></td>
<td>Specify the time limit in seconds for requests to be processed by the components between the <code>tESBProviderRequest</code> and the <code>tESBProviderResponse</code>.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong></td>
<td>the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>CORRELATION_ID</strong></td>
<td>the correlation ID by which chained service calls will be grouped. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td><strong>SECURITY_TOKEN</strong></td>
<td>the user identity information in the request header. This is a Flow variable and it returns an XML node.</td>
</tr>
<tr>
<td><strong>HEADERS_SOAP</strong></td>
<td>the headers of the SOAP request. This is a Flow variable and it returns all SOAP request headers.</td>
</tr>
<tr>
<td><strong>HEADERS_HTTP</strong></td>
<td>the headers of the HTTP request. This is a Flow variable and it returns all HTTP request headers.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

**Usage**

**Usage rule**

This component covers the possibility that a Talend Job can be wrapped as a service, with the ability to input a request to a service into a Job and return the Job result as a service response.

The tESBProviderResponse component can both deliver the payload of a SOAP message and also access the HTTP and SOAP headers of a service.

The tESBProviderRequest component should be used with the tESBProviderResponse component to provide a Job result as a response, in case of a request-response communication style.

When the SAML Token or the Service Registry is enabled in the service runtime options and if the SAML Token exists in the request header, the tESBProviderRequest component will get and store the SAML Token in the component variable for further use in the flow.

The tESBProviderRequest component will get the Correlation Value in the request header if it exists and stored it in the component variable. When the Business Correlation or the Service Registry is enabled in the service runtime options, the Correlation Value will also be added to the response. In this case, tESBProviderRequest will create a Correlation Value if it does not exist.

Note that the Service Registry option is only available if you subscribed to Talend Enterprise ESB solutions. For more information about how to set the runtime options, see the corresponding section in the Talend Studio User Guide.

**Dynamic settings**

Click the [+] button to add a row in the table and fill the Code field with a context variable to turn on or off the Keep listening option dynamically at runtime.

When a dynamic parameter is defined, the corresponding Keep listening option in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

**Limitation**

A JDK is required for this component to operate.
Scenario: Service sending a message without expecting a response

This scenario applies only to a Talend solution with ESB.

The Jobs, which are built upon the components under the ESB/Web Services family, act as the implementations of web services defined in the Services node of the Repository. They require the creation of and association with relevant services. For more information about services, see the related topics in the Talend Studio User Guide.

In this scenario, a provider Job and a consumer Job are needed. In the meantime, the related service should already exist in the Services node, with the WSDL URI being http://127.0.0.1.8088/ESB/provider/?WSDL, the port name being TEST_ProviderJobSoapBinding and the operation being invoke(anyType):anyType.

The provider Job consists of a tESBProviderRequest, a tXMLMap, and two tLogRow components.

• Drop the following components from the Palette onto the design workspace: a tESBProviderRequest, a tXMLMap, and two tLogRow.

• Double-click tESBProviderRequest_1 in the design workspace to display its Component view and set its Basic settings.

• Select Repository from the Property Type list and click the three-dot button to choose the service, to the granularity of port name and operation.
• Click OK.

• Click the three-dot button next to Edit schema to view the schema of tESBProviderRequest_1.

• Click OK.

• Connect tESBProviderRequest_1 to tLogRow_1.

• Double-click tLogRow_1 in the design workspace to display its Component view and set its Basic settings.

• Click the three-dot button next to Edit schema. and define the schema as follow.
• Connect \texttt{tLogRow\textsubscript{1}} to \texttt{tXMLMap\textsubscript{1}}.

• Connect \texttt{tXMLMap\textsubscript{1}} to \texttt{tLogRow\textsubscript{2}} and name this row as \textit{payload}.

• In the design workspace, double-click \texttt{tXMLMap\textsubscript{1}} to open the \textbf{Map Editor}.

• On the lower right part of the map editor, click the plus button to add one row to the \textit{payload} table and name this row as \textit{payload}.

• In the \textbf{Type} column of this \textit{payload} row, select \textbf{Document} as the data type. The corresponding XML root is added automatically to the top table on the right side which represents the output flow.

• In the \textit{payload} table, right-click \textit{root} to open the contextual menu.

• From the contextual menu, select \textbf{Create Sub-Element} and type in \textit{response} in the popup dialog box.

• Right-click the \textit{response} node and select \textbf{As loop element} from the contextual menu.

• Repeat this operation to create a sub-element \textit{request} of the \textit{root} node in the input table and set the \textit{request} node as loop element.

• Click the \textit{request} node in the input table and drop it to the \textbf{Expression} column in the row of the \textit{response} node in the output table.
• Click **OK** to validate the mapping and close the map editor.

• Double-click **tLogRow_2** in the design workspace to display its **Component** view and set its **Basic settings**.

• Click the three-dot button next to **Edit Schema** and define the schema as follow.
• Save the Job.

The consumer Job consists of a tFixedFlowInput, a tXMLMap, a tESBConsumer, and two tLogRow components.

• Drop the following components from the Palette onto the design workspace: a tFixedFlowInput, a tXMLMap, a tESBConsumer, and two tLogRow.

• Double-click tFixedFlowInput_1 in the design workspace to display its Component view and set its Basic settings.
• Edit the schema of the tFixedFlowInput_1 component.

![Schema of tFixedFlowInput_1](image)

• Click the plus button to add a new line of string type and name it `payloadString`.

• Click OK.

• In the Number of rows field, set the number of rows as 1.

• In the Mode area, select Use Single Table and input `world` in quotations into the Value field.

• Connect tFixedFlowInput_1 to tXMLMap_1.

• Connect tXMLMap_1 to tESBConsumer_1 and name this row as `payload`.

• In the design workspace, double-click tXMLMap_1 to open the Map Editor.

• In the output table, right-click the root node to open the contextual menu.

• From the contextual menu, select Create Sub-Element and type in `request` in the popup dialog box.

• Right-click the `request` node and select As loop element from the contextual menu.

• Click the `payloadstring` node in the input table and drop it to the Expression column in the row of the `request` node in the output table.
• Click **OK** to validate the mapping and close the **Map Editor**.

• Start the Provider Job. In the executing log you can see:

```
... web service [endpoint: http://127.0.0.1:8088/esb/provider] published ...
```

• In the **tESBConsumer_1 Component** view, set its **Basic settings**.

• Click the three-dot button next to the **Service Configuration** to open the editor.
• In the WSDL field, type in: http://127.0.0.1:8088/esb/provider?WSDL.

• Click the Refresh button to retrieve port name and operation name.

• Click OK.

• In the Basic settings of the tESBConsumer, set the Input Schema as follow:

• Set the Response Schema as follow:

• Set the Fault Schema as follow:

• Connect tESBConsumer_1 to tLogRow_1 and tLogRow_2.
• In the design workspace, double-click the **tLogRow_1** component to display its **Component** view and set its **Basic settings**.

![tLogRow_1](image)

- **Basic settings**
- **Advanced settings**
- **Dynamic settings**
- **View**
- **Documentation**
- **Validation Rules**

• Click the three-dot button next to **Edit Schema** and define the schema as follow:

![Edit Schema](image)

• In the Job Design, double-click **tLogRow_2** to display its **Component** view and set its **Basic settings**.

![tLogRow_2](image)

- **Basic settings**
- **Advanced settings**
- **Dynamic settings**
- **View**
- **Documentation**
- **Validation Rules**

• Click the three-dot button next to **Edit Schema** and define the schema as follow.

![Edit Schema](image)
• Save the Job.

• Run the provider Job. In the execution log you will see:

INFO: Setting the server’s publish address to be http://127.0.0.1:8088/esb/provider
2011-04-21 14:14:36.793:INFO:jetty-7.2.2.v20101205
2011-04-21 14:14:37.856:INFO::Started
SelectChannelConnector@127.0.0.1:8088
web service [endpoint: http://127.0.0.1:8088/esb/provider] published

• Run the consumer Job. In the execution log of the Job you will see:

Starting job CallProvider at 14:15 21/04/2011.
[statistics] connecting to socket on port 3942
[statistics] connected
TEST_ESBProvider2
TEST_ESBProvider2SoapBinding
|
[tLogRow_2] payloadString: <request>world</request>
(http://talend.org/esb/service/job)TEST_ESBProvider2
(http://talend.org/esb/service/job)TEST_ESBProvider2SoapBinding
invoke
[tLogRow_1] payload: null
[statistics] disconnected
Job CallProvider2 ended at 14:16 21/04/2011. [exit code=0]

• In the provider’s log you will see the trace log:

web service [endpoint: http://127.0.0.1:8088/esb/provider] published
[tLogRow_2] payload: <?xml version="1.0" encoding="UTF-8"?>
<request>world</request>
### world
[tLogRow_2] content: world
[tLogRow_3] payload: <?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://talend.org/esb/service/job">Hello, world!</response>
[statistics] disconnected
Job ESBProvider2 ended at 14:16 21/04/2011. [exit code=0]
**tESBProviderResponse**

Serves a Talend Job cycle result as a response message.

It acts as a service provider response builder at the end of each Talend Job cycle.

**tESBProviderResponse Standard properties**

These properties are used to configure tESBProviderResponse running in the Standard Job framework.

The Standard tESBProviderResponse component belongs to the ESB family.

*This component is relevant only when used with one of the Talend solutions with ESB, as it should be used with the Service Repository node and the Data Service creation related wizard(s).*

**Basic settings**

| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
• **View schema**: choose this option to view the schema only.  
• **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.  
Click **Sync columns** to retrieve the schema from the previous component connected in the Job. |
|---|---|

| **Built-in**: The schema is created and stored locally for this component only. Related topic: see [Talend Studio User Guide](#). |  |
| **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see [Talend Studio User Guide](#). |  |

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |
**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>The <strong>tESBProviderResponse</strong> component should only be used with the <strong>tESBProviderRequest</strong> component to provide a Job result as response for a web service provider, in case of a request-response communication style.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>A JDK is required for this component to operate.</td>
</tr>
</tbody>
</table>

**Scenario: Returning Hello world response**

This scenario applies only to a Talend solution with ESB.

The Jobs, which are built upon the components under the ESB/Web Services family, act as the implementations of web services defined in the **Services** node of the **Repository**. They require the creation of and association with relevant services. For more information about services, see the related topics in the [Talend Studio User Guide](#).

In this scenario, a provider Job and a consumer Job are needed. In the meantime, the related service should already exist in the **Services** node, with the WSDL URI being `http://127.0.0.1.8088/esb/provider/?WSDL`, the port name being **TEST_ProviderJobSoapBinding** and the operation being `invoke(anyType):anyType`.

The provider Job consists of a **tESBProviderRequest**, a **tESBProviderResponse**, a **tXMLMap**, and two **tLogRow** components.
• Drop the following components from the Palette onto the design workspace: a **tESBProviderRequest**, a **tESBProviderResponse**, a **tXMLMap**, and two **tLogRow**.

• In the design workspace, double-click **tESBProviderRequest_1** to display its **Component view** and set its **Basic settings**.

• Select **Repository** from the **Property Type** list and click the three-dot button to choose the service, to the granularity of port name and operation.

• Click **OK**.

• Click the three-dot button next to **Edit schema** to view its schema.
• Connect **tESBProviderRequest_1** to **tLogRow_1**.

• Double-click **tLogRow_1** to display its **Component** view and set its **Basic settings**.

• Click the three-dot button next to **Edit schema** and define the schema as follow.

• Connect **tLogRow_1** to **tXMLMap_1**.

• Connect **tXMLMap_1** to **tLogRow_2** and name this row as **payload**.

• In the design workspace, double-click **tXMLMap_1** to open the **Map Editor**.
• On the lower right part of the map editor, click the plus button to add one row to the \textit{payload} table and name this row as \textit{payload}.

• In the \textbf{Type} column of this \textit{payload} row, select \textbf{Document} as the data type. The corresponding XML root is added automatically to the top table on the right side which represents the output flow.

• In the \textit{payload} table, right-click \textit{root} to open the contextual menu.

• From the contextual menu, select \textbf{Create Sub-Element} and type in \textit{response} in the popup dialog box.

• Right-click the \textit{response} node and select \textbf{As loop element} from the contextual menu.

• Repeat this operation to create a sub-element \textit{request} of the \textit{root} node in the input table and set the \textit{request} node as loop element.

• Click the \textit{request} node in the input table and drop it to the \textbf{Expression} column in the row of the \textit{response} node in the output table.

• Click \textbf{OK} to validate the mapping and close the map editor.

• In the design workspace, double-click \texttt{tLogRow\_2} to display its \textbf{Component} view and set its \textbf{Basic settings}. 
• Click the three-dot button next to **Edit schema** and define the schema as follow.

• Connect **tLogRow_2** to **tESBProviderResponse_1**.

• In the design workspace, double-click **tESBProviderResponse_1** to open its **Component** view and set its **Basic settings**.

• Click the three-dot button next to **Edit schema** and define the schema as follow.
• Save the provider Job.

The consumer Job consists of a \texttt{tFixedFlowInput}, a \texttt{tXMLMap}, a \texttt{tESBConsumer}, and two \texttt{tLogRow} components.

• Drop the following components from the \texttt{Palette} onto the design workspace: a \texttt{tFixedFlowInput}, a \texttt{tXMLMap}, a \texttt{tESBConsumer}, and two \texttt{tLogRow}.

• Double-click \texttt{tFixedFlowInput\_1} in the design workspace to display its \texttt{Component} view and set its \texttt{Basic settings}.

• Click the three-dot button next to \texttt{Edit schema}. 
• Click the plus button to add a new line of string type and name it `payloadString`.
• Click **OK**.
• In the **Number of rows** field, set the number of rows as 1.
• In the **Mode** area, select **Use Single Table** and input `world` in quotations into the **Value** field.
• Connect **tFixedFlowInput** to **tXMLMap**.
• Connect **tXMLMap** to **tESBConsumer** and name this row as **payload**.
• In the design workspace, double-click **tXMLMap_1** to open the **Map Editor**.
• In the **payload** table, right-click **root** to open the contextual menu.
• From the contextual menu, select **Create Sub-Element** and type in `request` in the popup dialog box.
• Right-click the `request` node and select **As loop element** from the contextual menu.
• Click the `payloadstring` node in the input table and drop it to the **Expression** column in the row of the `request` node in the output table.
• Click OK to validate the mapping and close the **Map Editor**.

• Start the Provider Job. In the executing log you can see:

  ```
  ... web service [endpoint: http://127.0.0.1:8088/esb/provider] published ...
  ```

• In the **tESBConsumer_1 Component** view, set its **Basic settings**.

  ![tESBConsumer_1 Component view](image)

  ```
  Basic settings:
  Advanced settings:
  Dynamic settings:
  View:
  Documentation:
  Validation Rules:

  - Use Service Registry
  - Service configuration
  - Endpoint:
    `http://127.0.0.1:8088/esb/provider/?WSDL`

  - Input Schema: **Built-In**
  - Response Schema: **Built-In**
  - Fault Schema: **Built-In**

  - Use Service Locator
  - Use Service Activity Monitor
  - Use Authentication
  - Use Business Correlation
  - Die on error
  ```

• Click the three-dot button next to the **Service Configuration** to open the editor.

  ![Service Configuration editor](image)

  ```
  Configure component with Web Service operation

  WSDL: `http://127.0.0.1:8088/esb/provider/?WSDL`
  Port Name: `TEST_ProviderJobSoapBinding`
  Operation: `invoke(anyType):anyType(string):string`

  - Populate schema to repository on finish
  ```

• In the **WSDL** field, type in: `http://127.0.0.1:8088/esb/provider/?WSDL`

• Click the Refresh button to retrieve port name and operation name.
• Click OK.

• In the Basic settings of the tESBConsumer, set the Input Schema as follows:

![Input Schema](image)

• Set the Response Schema as follows:

![Response Schema](image)

• Set the Fault Schema as follows:

![Fault Schema](image)

• Connect tESBConsumer_1 to tLogRow_1 and tLogRow_2.

• In the design workspace, double-click tLogRow_1 to display its Component view and set its Basic settings.

![Component View](image)

• Click the three-dot button next to Edit Schema and define the schema as follow.
• In the Job Design, double-click tLogRow_2 to display its Component view and set its Basic settings.

• Click the three-dot button next to Edit Schema and define the schema as follow:

• Save the consumer Job.

• Run the provider Job. In the execution log you will see:

  2011-04-21 15:28:26.874:INFO::jetty-7.2.2.v20101205
  SelectChannelConnector@127.0.0.1:8088
  web service [endpoint: http://127.0.0.1:8088/esb/provider] published

• Run the consumer Job. In the execution log of the Job you will see:

  Starting job CallProvider at 15:29 21/04/2011.
  [statistics] connecting to socket on port 3690
  [statistics] connected
  TEST_ProviderJob
  TEST_ProviderJobSoapBinding
  | {http://talend.org/esb/service/job}TEST_ProviderJob
  | {http://talend.org/esb/service/job}TEST_ProviderJobSoapBinding
  invoke
In the provider’s log you will see the trace log:

```
[tLogRow_1] payload: <?xml version="1.0" encoding="UTF-8"?>
<request>world</request>
### world
[tLogRow_2] content: world
[tLogRow_3] payload: <?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://talend.org/esb/service/job">Hello, world!</response>
[statistics] disconnected
Job ProviderJob ended at 15:29 21/04/2011. [exit code=0]
```
tEXABulkExec

Imports data into an EXASolution database table using the IMPORT command provided by the EXASolution database in a fast way.

The import will be cancelled after a configurable number of records fail to import. Erroneous records can be sent to a log table in the same database or to a local log file.

**tEXABulkExec Standard properties**

These properties are used to configure tEXABulkExec running in the Standard Job framework.

The Standard tEXABulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and from the list displayed select the relevant connection component to reuse the connection details you have already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td></td>
<td>For an example about how to share a database connection across Job levels, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>• Repository: Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

<p>| Host | Enter the host or host list of the EXASol database servers. EXASol can run in a cluster environment. The valid value can be a simple IP address (for example, 172.16.173.128), an IP range list (for example, 172.16.173.128..130 that represents three servers 172.16.173.128, 172.16.173.129, and 172.16.173.130), or |</p>
<table>
<thead>
<tr>
<th>Port</th>
<th>Enter the listening port number of the EXASolution database cluster.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>Enter the name of the schema you want to use.</td>
</tr>
<tr>
<td>User and Password</td>
<td>Enter the user authentication data to access the EXASolution database. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Enter the name of the table to be written.</td>
</tr>
<tr>
<td>Note:</td>
<td>Typically the table names are stored in uppercase. If you need mixed case identifiers, you have to enter the name in double quotes. For example, &quot;\TEST_data_LOAD&quot;.</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations before running the import:</td>
</tr>
<tr>
<td>• None: No operation is carried out.</td>
<td></td>
</tr>
<tr>
<td>• Drop and create table: The table is removed and created again.</td>
<td></td>
</tr>
<tr>
<td>• Create table: The table does not exist and gets created.</td>
<td></td>
</tr>
<tr>
<td>• Create table if not exists: The table is created if it does not exist.</td>
<td></td>
</tr>
<tr>
<td>• Truncate table: The table content is deleted. You do not have the possibility to rollback the operation.</td>
<td></td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td>• Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>• Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td>The columns in the schema must be in the same order as they are in the CSV file. It is not necessary to fill all columns of the defined table unless the use case or table definition expects that.</td>
</tr>
</tbody>
</table>
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Advanced settings

**Additional JDBC Parameters**

Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair, for example, `encryption=1;clientname=Talend`.

This field is not available if the **Use an existing connection** check box is selected.

**Column Formats**

Specify the format for Date and numeric columns if the default can not be applied.

- **Column**: The cells in this column are automatically filled with the defined schema column names.
- **Has Thousand Delimiters**: Select this check box if the value of the corresponding numeric column (only for numeric column) in the file contains thousand separators.
- **Alternative Format**: Specify the necessary format as String value if a special format is expected. The necessary format will be created from the schema column length and precision. For more information about format models, see **EXASolution User Manual**.

**Source table columns**

If the source is a database, configure the mapping between the source columns and the target columns in this table.

Specifically configuring the mapping is optional. If you set nothing here, it is assumed that the source table has the same structure as the target table.

- **Column**: The schema column in the target table.
- **Source column name**: The name of the column in the source table.

**Column Separator**

Enter the separator for the columns of a row in the local file.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Delimiter</td>
<td>Enter the delimiter that encapsulates the field content in the local file.</td>
</tr>
<tr>
<td>Row Separator</td>
<td>Enter the char used to separate the rows in the local file.</td>
</tr>
<tr>
<td>Null representation</td>
<td>Enter the string that represents a NULL value in the local file. If not specified, NULL values are represented as the empty string.</td>
</tr>
<tr>
<td>Skip rows</td>
<td>Enter the number of rows (for example, header or any other prefix rows) to be omitted.</td>
</tr>
<tr>
<td>Encoding</td>
<td>Enter the character set used in the local file. By default, it is UTF8.</td>
</tr>
<tr>
<td>Trim column values</td>
<td>Specify whether spaces are deleted at the border of CSV columns.</td>
</tr>
<tr>
<td></td>
<td>• <strong>No trim</strong>: no spaces are trimmed.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Trim</strong>: spaces from both left and right sides are trimmed.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Trim only left</strong>: spaces from only the left side are trimmed.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Trim only right</strong>: spaces from only the right side are trimmed.</td>
</tr>
<tr>
<td>Default Date Format</td>
<td>Specify the format for datetime values. By default, it is YYYY-MM-DD.</td>
</tr>
<tr>
<td>Default Timestamp Format</td>
<td>Specify the timestamp format used. By default, it is YYYY-MM-DD HH24:MI:SS.FF3.</td>
</tr>
<tr>
<td>Thousands Separator</td>
<td>Specify the character used to separate thousand groups in a numeric text value. In the numeric format, the character will be applied to the placeholder G. If the text values contain this char, you have to configure it also in the Column Formats table.</td>
</tr>
<tr>
<td></td>
<td>Note that this setting affects the connection property NLS_NUMERIC_CHARACTERS that defines the decimal and group characters used for representing numbers.</td>
</tr>
<tr>
<td>Decimal Separator</td>
<td>Specify the character used to separate the integer part of a number from the fraction. In the numeric format, the character will be applied to the placeholder D.</td>
</tr>
<tr>
<td></td>
<td>Note that this setting affects the connection property NLS_NUMERIC_CHARACTERS that defines the decimal and group characters used for representing numbers.</td>
</tr>
<tr>
<td>Minimal number errors to reject the transfer</td>
<td>Specify the maximum number of invalid rows allowed during the data loading process. For example, the value 2 means the loading process will stop if the third error occurs.</td>
</tr>
<tr>
<td>Log Error Destination</td>
<td>Specify the location where error messages will be stored.</td>
</tr>
<tr>
<td></td>
<td>• <strong>No Logging</strong>: error messages will not be saved.</td>
</tr>
</tbody>
</table>
• **Local Log File**: error messages will be stored in a specified local file.

• **Local Error Log File**: specify the path to the local file that stores error messages.

• **Add current timestamp to log file name (before extension)**: select this check box to add the current timestamp before the extension of the file name for identification reasons in case you use the same file multiple times.

• **Logging Table**: error messages will be stored in a specified table. The table will be created if it does not exist.

• **Error Log Table**: enter the name of the table that stores error messages.

• **Use current timestamp to build log table**: select this check box to use the current timestamp to build the log table for identification reasons in case you use the same table multiple times.

<table>
<thead>
<tr>
<th>Transfer files secure</th>
<th>Select this check box to transfer the file over HTTPS instead of HTTP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_DELETED</strong>: the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>FILENAME</strong>: the name of the file processed. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_LOG_FILE</strong>: the path to the local log file. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is usually used as a standalone component.</th>
</tr>
</thead>
</table>

**Dynamic settings**
Click the [+| button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Settings for different sources of import data

The settings for this component change depending on the source of your import data.

The component handles data coming from any of the following sources:

- Local file
- Remote file
- EXASol database
- Oracle database
- JDBC-compliant database

**Local file**
The local file is not transferred by uploading the file. Instead, the driver establishes a (secure) web service that sends the URL to the database, and the database retrieves the file from this local web service. Because the port of this service cannot be explicitly defined, this method requires a transparent network between the local Talend Job and the EXASolution database.

<table>
<thead>
<tr>
<th><strong>File name</strong></th>
<th>Specify the path to the local file to be imported.</th>
</tr>
</thead>
</table>
Remote file

This method works with a file that is accessible on a server through the following protocols: SCP, SFTP, FTP, HTTP, or HTTPS.

<table>
<thead>
<tr>
<th>Use predefined connection</th>
<th>It is possible, via the SQL interface, to set up a named connection in the EXASolution database itself. Select this option if you want to use such a connection, and provide its name. To know what connections are available, look at the table SYS.EXA_DBA_CONNECTIONS in the database. The connection must contain a URL with one of the following protocols: SCP, SFTP, FTP, HTTP, or HTTPS. The URL must not contain the file name. The file name is always dynamic and must be provided by the component configuration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote file server URL</td>
<td>Specify the URL to the file server, without the file name itself.</td>
</tr>
<tr>
<td>File name</td>
<td>Specify the name of the file you want to fetch from the server.</td>
</tr>
<tr>
<td>Query parameters</td>
<td>If the web service depends on query parameters, specify them here. For example, if you want to get a file from an HDFS file system via the web service, you need to add some additional parameters such as open=true.</td>
</tr>
<tr>
<td>Use user authentication</td>
<td>Select this check box if you want to use Basic Authentication when connecting to the web server.</td>
</tr>
<tr>
<td>Remote user and Remote users password</td>
<td>Enter the user name and password need to access the web server.</td>
</tr>
</tbody>
</table>

EXASol database

An EXASolution database can also serve as a remote source for the data. The source can be a table or a specific query.

<table>
<thead>
<tr>
<th>Use predefined connection</th>
<th>It is possible, via the SQL interface, to set up a named connection in the EXASolution database itself. Select this option if you want to use such a connection, and provide its name. To know what connections are available, look at the table SYS.EXA_DBA_CONNECTIONS in the database. The username and password must by provided by the component and not as part of the predefined connection.</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXASol database host</td>
<td>Specify the host of the remote EXASolution database. This field can also be used to access a cluster.</td>
</tr>
<tr>
<td>Use self defined query</td>
<td>Select this check box if you want to use a specific query to get the data. This method is preferred if, for example, your data needs to be filtered (using a where condition), joined or converted.</td>
</tr>
<tr>
<td>Source query</td>
<td>If you want to use a specific query, enter the query in this field.</td>
</tr>
<tr>
<td>Database or schema</td>
<td>If you are not using a specific query, enter the schema name for the source table in this field.</td>
</tr>
</tbody>
</table>
**Source table**
If you are not using a specific query, enter the table name in this field. The mapping between the source table columns and the target table columns (schema columns) can be set in the advanced settings.

**Use user authentication**
Select this check box if you want to use Basic Authentication when connecting to the source database.

**Remote user and Remote users password**
Enter the user name and password needed to access the source database.

---

**Oracle database**
An Oracle database can also serve as remote source for the data. Access to an Oracle database requires an Enterprise license for the EXASolution database and does not work with the free edition. The source can be a table or a specific query.

**Use predefined connection**
It is possible, via the SQL interface, to set up a named connection in the EXASolution database itself. Select this option if you want to use such a connection, and provide its name.
To know what connections are available, look at the table SYS.EXA_DBA_CONNECTIONS in the database.
The username and password must by provided by the component and not as part of the predefined connection.

**Oracle database URL**
Specify the JDBC URL to the Oracle database.

**Use self defined query**
Select this check box if you want to use a specific query to get the data.
This method is preferred if, for example, your data needs to be filtered (using a *where* condition), joined or converted.

**Source query**
If you want to use a specific query, enter the query in this field.

**Database or schema**
If you are not using a specific query, enter the schema name for the source table in this field.

**Source table**
If you are not using a specific query, enter the table name in this field.
The mapping between the source table columns and the target table columns (schema columns) can be set in the advanced settings.

**Use user authentication**
Select this check box if you want to use Basic Authentication when connecting to the source database.

**Remote user and Remote users password**
Enter the user name and password needed to access the source database.

---

**JDBC-compliant database**
The free edition of the EXASolution database supports MySQL and PostgreSQL databases, and others are available in the Enterprise edition. The source can be table or a self defined query.

Nearly all enterprise-grade databases provide a JDBC interface.
It is possible, via the SQL interface, to set up a named connection in the EXASolution database itself. Select this option if you want to use such a connection, and provide its name.

To know what connections are available, look at the table SYS.EXA_DBA_CONNECTIONS in the database.

The username and password must by provided by the component and not as part of the predefined connection.

Specify the JDBC URL to the source database.

Select this check box if you want to use a specific query to get the data.

This method is preferred if, for example, your data needs to be filtered (using a `where` condition), joined or converted.

If you want to use a specific query, enter the query in this field.

If you are not using a specific query, enter the schema name for the source table in this field.

If you are not using a specific query, enter the table name in this field.

The mapping between the source table columns and the target table columns (schema columns) can be set in the advanced settings.

Select this check box if you want to use Basic Authentication when connecting to the source database.

Enter the user name and password needed to access the source database.

---

**Scenario: Importing data into an EXASolution database table from a local CSV file**

This scenario describes a Job that writes employee information into a CSV file, then loads the data from this local file into a newly created EXASolution database table using the `tEXABulkExec` component, and finally retrieves the data from the table and displays it on the console.
Dropping and linking the components

Procedure

1. Create a new Job and add the following components by typing their names in the design workspace or dropping them from the Palette: a `tFixedFlowInput` component, a `tFileOutputDelimited` component, a `tEXABulkExec` component, a `tEXAInput` component, and a `tLogRow` component.

2. Connect the `tFixedFlowInput` component to the `tFileOutputDelimited` component using a Row > Main connection.

3. Do the same to connect the `tEXAInput` component to the `tLogRow` component.

4. Connect the `tFixedFlowInput` component to the `tEXABulkExec` component using a Trigger > On Subjob Ok connection.

5. Do the same to connect the `tEXABulkExec` component to the `tEXAInput` component.

Configuring the components

Preparing the source data

Procedure

1. Double-click the `tFixedFlowInput` component to open its Basic settings view.
2. Click the [...] button next to Edit schema to open the [Schema] dialog box.

3. Click the [+] button to add six columns: EmployeeID of the Integer type, EmployeeName, OrgTeam and JobTitle of the String type, OnboardDate of the Data type with the yyyy-MM-dd date pattern, and MonthSalary of the Double type.

4. Click OK to close the dialog box and accept schema propagation to the next component.

5. In the Mode area, select Use Inline Content (delimited file) and enter the following employee data in the Content field.

```
12000;James;Dev Team;Developer;2008-01-01;15000.01
12001;Jimmy;Dev Team;Developer;2008-11-22;13000.11
12002;Herbert;QA Team;Tester;2008-05-12;12000.22
12003;Harry;Doc Team;Technical Writer;2009-03-10;12000.33
12004;Ronald;QA Team;Tester;2009-06-20;12500.44
12005;Mike;Dev Team;Developer;2009-10-15;14000.55
12006;Jack;QA Team;Tester;2009-03-25;13500.66
12007;Thomas;Dev Team;Developer;2010-02-20;16000.77
12008;Michael;Dev Team;Developer;2010-07-15;14000.99
12009;Peter;Doc Team;Technical Writer;2011-02-10;12500.99
```
6. Double-click the tFileOutputDelimited component to open its Basic settings view.

7. In the File Name field, specify the file into which the input data will be written. In this example, it is "E:/employee.csv".

8. Click Advanced settings to open the Advanced settings view of the tFileOutputDelimited component.

9. Select the Advanced separator (for numbers) check box and in the Thousands separator and Decimal separator fields displayed, specify the separators for thousands and decimal. In this example, the default values "," and "." are used.

Loading the source data into a newly created EXASolution database table

Procedure

1. Double-click the tEXABulkExec component to open its Basic settings view.
2. Fill in the **Host**, **Port**, **Schema**, **User** and **Password** fields with your EXASolution database connection details.

3. In the **Table** field, enter the name of the table into which the source data will be written. In this example, the target database table is named "employee" and it does not exist.

4. Select **Create table** from the **Action on table** list to create the specified table.

5. In the **Source** area, select **Local file** as the source for the input data, and then specify the file that contains the source data. In this example, it is "E:/employee.csv".

6. Click the [...] button next to **Edit schema** to open the [Schema] dialog box and define the schema, which should be the same as that of the **tFixedFlowInput** component.

   Then click **OK** to validate these changes and close the dialog box.

7. Click **Advanced settings** to open the **Advanced settings** view of the **tEXABulkExec** component.

   In the **Column Formats** table, for the two numeric fields **EmployeeID** and **MonthSalary**, select the corresponding check boxes in the **Has Thousand Delimiters** column, and then define their format.
model strings in the corresponding fields of the **Alternative Format** column. In this example, "99G999" for EmployeeID and "99G999D99" for MonthSalary.

9. Make sure that the **Thousands Separator** and **Decimal Separator** fields have values identical to those of the **tFileOutputDelimited** component and keep the default settings for the other options.

**Retrieving data from the EXASolution database table**

**Procedure**

1. Double-click the **tEXAInput** component to open its **Basic settings** view.

![tEXAInput component](image)

- **Host name**: "192.168.31.13"
- **Port**: "8563"
- **Username**: "sys"
- **Password**: ********
- **Table Name**: "employee"
- **Query**: ```
SELECT employee.EmployeeID, employee.EmployeeName, employee.OrgTeam, employee.JobTitle, employee.OnboardDate, employee.MonthSalary
FROM employee
```  

2. Fill in the **Host name**, **Port**, **Schema name**, **Username** and **Password** fields with your EXASolution database connection details.

3. In the **Table Name** field, enter the name of the table from which the data will be retrieved. In this example, it is "employee".

4. Click the [...] button next to **Edit schema** to open the [Schema] dialog box and define the schema, which should be the same as that of the **tFixedFlowInput** component.

Then click **OK** to close the dialog box and accept schema propagation to the next component.

5. Click the **Guess Query** button to fill the **Query** field with the following auto-generated SQL statement that will be executed on the specified table.

```sql
SELECT employee.EmployeeID, employee.EmployeeName, employee.OrgTeam, employee.JobTitle, employee.OnboardDate, employee.MonthSalary
FROM employee
```

6. Double-click the **tLogRow** component to open its **Basic settings** view.
7. In the **Mode** area, select the **Table (print values in cells of a table)** option for better readability of the output.

### Saving and executing the Job

**Procedure**

1. Press `Ctrl + S` to save the Job.
2. Press `F6` to execute the Job.

```plaintext
[statistics] connecting to socket on port 3680
[statistics] connected

<table>
<thead>
<tr>
<th>EmployeeID</th>
<th>EmployeeName</th>
<th>OrgTeam</th>
<th>JobTitle</th>
<th>OnboardingDate</th>
<th>MonthSalary</th>
</tr>
</thead>
<tbody>
<tr>
<td>12000</td>
<td>James</td>
<td>Dev Team</td>
<td>Developer</td>
<td>2008-01-01</td>
<td>15000.01</td>
</tr>
<tr>
<td>12001</td>
<td>Jimmy</td>
<td>Dev Team</td>
<td>Developer</td>
<td>2008-11-22</td>
<td>13000.11</td>
</tr>
<tr>
<td>12002</td>
<td>Herbert</td>
<td>QA Team</td>
<td>Tester</td>
<td>2008-05-12</td>
<td>12000.22</td>
</tr>
<tr>
<td>12003</td>
<td>Harry</td>
<td>Doc Team</td>
<td>Technical Writer</td>
<td>2009-03-10</td>
<td>12000.33</td>
</tr>
<tr>
<td>12004</td>
<td>Ronald</td>
<td>QA Team</td>
<td>Tester</td>
<td>2009-06-20</td>
<td>12500.44</td>
</tr>
<tr>
<td>12005</td>
<td>Mike</td>
<td>Dev Team</td>
<td>Developer</td>
<td>2009-10-15</td>
<td>14000.55</td>
</tr>
<tr>
<td>12006</td>
<td>Jack</td>
<td>QA Team</td>
<td>Tester</td>
<td>2009-03-25</td>
<td>13500.66</td>
</tr>
<tr>
<td>12007</td>
<td>Thomas</td>
<td>Dev Team</td>
<td>Developer</td>
<td>2010-02-20</td>
<td>16000.77</td>
</tr>
<tr>
<td>12008</td>
<td>Michael</td>
<td>Dev Team</td>
<td>Developer</td>
<td>2010-07-15</td>
<td>14000.88</td>
</tr>
<tr>
<td>12009</td>
<td>Peter</td>
<td>Doc Team</td>
<td>Technical Writer</td>
<td>2011-02-10</td>
<td>12500.99</td>
</tr>
</tbody>
</table>

[statistics] disconnected
```

As shown above, the employee data is written into the specified EXASolution database table and is then retrieved and displayed on the console.
tEXAClose

Closes an active connection to an EXASolution database instance to release the occupied resources.

**tEXAClose Standard properties**

These properties are used to configure tEXAClose running in the Standard Job framework.

The Standard tEXAClose component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the <strong>tEXAConnection</strong> component that opens the connection you need to close from the list.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

### Global Variables

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.
**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other EXASolution components, especially with the tEXAConnection and tEXAClose components.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.  

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.  
For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

**Related scenario**

No scenario is available for the Standard version of this component yet.
tEXACommit

Validates the data processed through the Job into the connected EXASolution database. Using a unique connection, this component commits in one go a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

**tEXACommit Standard properties**

These properties are used to configure tEXACommit running in the Standard Job framework. The Standard tEXACommit component belongs to the Databases family. The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the tEXAConnection component for which you want the commit action to be performed.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default and it allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection after the component has performed its task. <strong>Warning:</strong> If you want to use a Row &gt; Main connection to link tEXACommit to your Job, your data will be committed row by row. In this case, do not select the Close Connection check box or your connection will be closed before the end of your first row commit.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After |
variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other EXASolution components, especially with the tEXAConnection and tEXARollback components.</th>
</tr>
</thead>
</table>

| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

Related scenario

For a similar scenario using other database, see Scenario: Inserting data in mother/daughter tables on page 2476.
tEXACo\nnection

Opens a connection to an EXASolution database instance that can then be reused by other EXASolution components.

tEXACo\nnection Standard properties

These properties are used to configure tEXACo\nnection running in the Standard Job framework.
The Standard tEXACo\nnection component belongs to the Databases and the ELT families.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Enter the host or host list of the EXASol database servers. EXASol can run in a cluster environment. The valid value can be a simple IP address (for example, 172.16.173.128), an IP range list (for example, 172.16.173.128..130 that represents three servers 172.16.173.128, 172.16.173.129, and 172.16.173.130), or a comma-separated host list (for example, server1,server2,server3) of the EXASolution database cluster.</td>
</tr>
<tr>
<td>Port</td>
<td>Enter the listening port number of the EXASolution database cluster.</td>
</tr>
<tr>
<td>Schema</td>
<td>Enter the name of the schema you want to use.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Enter the user authentication data to access the EXASolution database. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>
| Use or register a shared DB Connection | Select this check box to share your database connection or fetch a database connection shared by a parent or
child Job, and in the **Shared DB Connection Name** field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the **tRunJob** component. Using a shared connection together with a **tRunJob** component with either of these two options enabled will cause your Job to fail.

### Advanced settings

| **Auto Commit** | Select this check box to commit any changes to the database automatically upon the transaction. With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component. Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component. |
| **Additional JDBC Parameters** | Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair, for example, `encryption=1;clientname=Talend`. This field is not available if the **Use an existing connection** check box is selected. |
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

### Global Variables

| **Global Variables** | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

| Usage rule | This component is more commonly used with other EXASolution components, especially with the tEXACcommit and tEXARollback components. |

### Related scenario

For a similar scenario using other database, see Scenario: Inserting data in mother/daughter tables on page 2476.
tEXAInput

Retrieves data from an EXASolution database based on a query with a strictly defined order which corresponds to the schema definition, and passes the data to the next component.

### tEXAInput Standard properties

These properties are used to configure tEXAInput running in the Standard Job framework.

The Standard tEXAInput component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components on page 598](#).

#### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Either Built-In or Repository.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see [Talend Studio User Guide](#).

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and from the list displayed select the relevant connection component to reuse the connection details you have already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td>1.</td>
<td>In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
</tr>
</tbody>
</table>
2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host name</strong></td>
<td>Enter the host or host list of the EXASol database servers. EXASol can run in a cluster environment. The valid value can be a simple IP address (for example, 172.16.173.128), an IP range list (for example, 172.16.173.128..130 that represents three servers 172.16.173.128, 172.16.173.129, and 172.16.173.130), or a comma-separated host list (for example, server1,server2,server3) of the EXASolution database cluster.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Enter the listening port number of the EXASolution database cluster.</td>
</tr>
<tr>
<td><strong>Schema name</strong></td>
<td>Enter the name of the schema you want to use.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>Enter the user authentication data to access the EXASolution database. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Schema</strong> and <strong>Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. - <strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>. - <strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the Repository type, three options are available: - <strong>View schema</strong>: choose this option to view the schema only. - <strong>Change to built-in property</strong>: choose this option to change the schema to Built-in for local changes. - <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the <strong>Repository Content</strong> window.</td>
</tr>
</tbody>
</table>
## Table Name
Enter the name of the table to be queried.

## Query Type and Query
Enter the database query, paying particularly attention to the proper sequence of the fields in order to match the schema definition.

## Guess Query
Click the button to generate the query that corresponds to the table schema in the Query field.

## Guess schema
Click the button to retrieve the schema from the table.

### Advanced settings

#### Change fetch size
Select this check box to change the fetch size which specifies the amount of resultset data sent during one single communication step with the database. In the **Fetch size** field displayed, you need to enter the size in KB.

#### Additional JDBC parameters
Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair, for example, `encryption=1;clientname=Talend`.

This field is not available if the **Use an existing connection** check box is selected.

#### Trim all the String/Char columns
Select this check box to remove leading and trailing whitespaces from all the String/Char columns.

#### Trim column
Select the check box in the **Trim** column to remove leading and trailing whitespaces from the corresponding field.

This table is not available if the **Trim all the String/Char columns** check box is selected.

#### tStatCatcher Statistics
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

#### Global Variables

- **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

- **QUERY**: the query statement being processed. This is a Flow variable and it returns a string.

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide.*

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is usually used as a start component of a Job or Subjob and it needs an output link.</th>
</tr>
</thead>
</table>

**Dynamic settings**

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489.

For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide.*

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in *Talend Help Center* (https://help.talend.com).

### Related scenario

For a related scenario, see *Scenario: Importing data into an EXASolution database table from a local CSV file* on page 890.

For similar scenarios using other databases, see:
tEXAOoutput

Writes, updates, modifies or deletes data in an EXASolution database by executing the action defined on the table and/or on the data in the table, based on the flow incoming from the preceding component.

**tEXAOoutput Standard properties**

These properties are used to configure tEXAOoutput running in the Standard Job framework.

The Standard tEXAOoutput component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Either <strong>Built-In</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

Use an existing connection

Select this check box and from the list displayed select the relevant connection component to reuse the connection details you have already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component.
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

| **Host** | Enter the host or host list of the EXASol database servers. EXASol can run in a cluster environment. The valid value can be a simple IP address (for example, 172.16.173.128), an IP range list (for example, 172.16.173.128..130 that represents three servers 172.16.173.128, 172.16.173.129, and 172.16.173.130), or a comma-separated host list (for example, server1,server2,server3) of the EXASolution database cluster. |
| **Port** | Enter the listening port number of the EXASolution database cluster. |
| **Schema name** | Enter the name of the schema you want to use. |
| **Username** and **Password** | Enter the user authentication data to access the EXASolution database. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Table** | Enter the name of the table to be written. Note that only one table can be written at a time. |
| **Action on table** | On the table defined, you can perform one of the following operations:

- **None**: No operation is carried out.
- **Drop and create table**: The table is removed and created again.
- **Create table**: The table does not exist and gets created.
- **Create table if does not exist**: The table is created if it does not exist.
- **Drop table if exists and create**: The table is removed if it already exists and created again.
- **Clear table**: The table content is deleted.
- **Truncate table**: The table content is deleted. You don not have the possibility to rollback the operation. |
| **Action on data** | On the data of the table defined, you can perform:

- **Insert**: Add new entries to the table. If duplicates are found, Job stops.
- **Update**: Make changes to existing entries |
**Insert or update**: Insert a new record. If the record with the given reference already exists, an update would be made.

**Update or insert**: Update the record with the given reference. If the record does not exist, a new record would be inserted.

**Delete**: Remove entries corresponding to the input flow.

❗️ **Warning**: You must specify at least one column as a primary key on which the **Update** and **Delete** operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the update and delete operations. To do that: Select the **Use field options** check box and then in the **Key in update** column, select the check boxes next to the column name on which you want to base the update operation. Do the same in the **Key in delete** column for the deletion operation.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

**Die on error**

Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a **Row > Reject** link.

### Advanced settings

**Use commit control**

Select this box to display the **Commit every** field in which you can define the number of rows to be processed before committing.

**Additional JDBC parameters**

Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair, for example, `encryption=1;clientname=Talend`.

This field is not available if the **Use an existing connection** check box is selected.

**Additional Columns**

This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.

- **Name**: Enter the name of the column to be modified or inserted.
- **SQL expression**: Enter the SQL expression to be executed to modify or insert data in the corresponding columns.
- **Position**: Select **Before**, **After** or **Replace**, depending on the action to be carried out on the reference column.
- **Reference column**: Type in a column of reference that can be used to place or replace the new or altered column.

**Use field options**

Select this check box to customize a request for the corresponding column, particularly if multiple actions are being carried out on the data.

- **Key in update**: Select the check box for the corresponding column based on which the data is updated.
- **Key in delete**: Select the check box for the corresponding column based on which the data is deleted.
- **Updatable**: Select the check box if the data in the corresponding column can be updated.
### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</td>
<td></td>
</tr>
<tr>
<td><strong>NB_LINE_UPDATED</strong>: the number of rows updated. This is an After variable and it returns an integer.</td>
<td></td>
</tr>
<tr>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
<td></td>
</tr>
<tr>
<td><strong>NB_LINE_DELETED</strong>: the number of rows deleted. This is an After variable and it returns an integer.</td>
<td></td>
</tr>
<tr>
<td><strong>NB_LINE_REJECTED</strong>: the number of rows rejected. This is an After variable and it returns an integer.</td>
<td></td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
<td></td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in an EXASolution database. It also allows you to create a reject flow using a <strong>Row &gt; Rejects</strong> link to filter data in error. For a related scenario, see <em>Scenario: Retrieving data in error with a Reject link</em> on page 2529.</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Related scenario

For similar scenarios using other databases, see:

- Scenario: Inserting a column and altering data using tMysqlOutput on page 2521.
tEXARollback

Cancels the transaction commit in the connected EXASolution database.
It allows you to roll back any changes made in the EXASolution database to prevent partial transaction commit if an error occurs.

**tEXARollback Standard properties**

These properties are used to configure tEXARollback running in the Standard Job framework.
The Standard tEXARollback component belongs to the Databases family.
The component in this framework is generally available.

*Note:*
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the <strong>tEXAConnection</strong> component for which you want the rollback action to be performed.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default and it allows you to close the database connection once the rollback is done. Clear this check box to continue to use the selected connection after the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other EXASolution components, especially with the tEXAConnection and tEXACommit components.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Related Scenario

For a similar scenario using other database, see Scenario: Rollback from inserting data in mother/daughter tables on page 2480.
tEXARow

Executes SQL queries on an EXASolution database.

Depending on the nature of the query and the database, tEXARow acts on the actual structure of the database, or indeed the data, although without modifying them. The Row suffix indicates that it is used to channel a flow in a Job although it does not produce any output data.

**tEXARow Standard properties**

These properties are used to configure tEXARow running in the Standard Job framework.

The Standard tEXARow component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Either Built-In or Repository.</td>
</tr>
<tr>
<td></td>
<td>• Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>• Repository: Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and from the list displayed select the relevant connection component to reuse the connection details you have already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong> When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
</tbody>
</table>
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

| **Host** | Enter the host or host list of the EXASol database servers. EXASol can run in a cluster environment. The valid value can be a simple IP address (for example, 172.16.173.128), an IP range list (for example, 172.16.173.128..130 that represents three servers 172.16.173.128, 172.16.173.129, and 172.16.173.130), or a comma-separated host list (for example, server1,server2,server3) of the EXASolution database cluster. |
| **Port** | Enter the listening port number of the EXASolution database cluster. |
| **Schema name** | Enter the name of the schema you want to use. |
| **Username and Password** | Enter the user authentication data to access the EXASolution database. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
- **Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.  
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.  
Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
| **Table Name** | Enter the name of the table to be processed. |
| **Query Type** | Either Built-In or Repository.
- **Built-In**: Enter the query manually or with the help of the SQLBuilder.
- **Repository**: Select the appropriate query from the Repository. The Query field is then completed automatically.

| **Guess Query** | Click the **Guess Query** button to generate the query that corresponds to the table schema in the Query field. |
| **Query** | Enter the database query paying particularly attention to the proper sequence of the fields in order to match the schema definition. |
| **Die on error** | Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row > Reject link. |

### Advanced settings

| **Additional JDBC parameters** | Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair, for example, `encryption=1;clientname=Talend`. This field is not available if the **Use an existing connection** check box is selected. |
| **Propagate QUERY’s recordset** | Select this check box to insert the query results in one of the flow columns. Select the particular column from the use column list. |
| **Use PreparedStatement** | Select this check box to use prepared statements and in the Set PreparedStatement Parameters table displayed, add as many parameters as needed and set the following attributes for each parameter:

  - **Parameter Index**: enter the index of the prepared statement parameter.
  - **Parameter Type**: click in the cell and select the type of the parameter from the list.
  - **Parameter Value**: enter the value of the parameter. |
| **Commit every** | Enter the number of rows to be included in each batch before the data is written. This option guarantees the quality of the transaction (although there is no rollback option) and improves performance. |
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |
### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>QUERY</strong>: the query statement being processed. This is a Flow variable and it returns a string.</td>
<td></td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
<td></td>
</tr>
<tr>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
<td></td>
</tr>
<tr>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
<td></td>
</tr>
<tr>
<td>For further information about variables, see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage rule</strong>: This component offers query flexibility as it covers all possible SQL query requirements.</td>
<td></td>
</tr>
<tr>
<td><strong>Limitation</strong>: Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the <strong>Install</strong> button on the <strong>Component</strong> tab view. You can also find out and add all missing JARs easily on the <strong>Modules</strong> tab in the <strong>Integration</strong> perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
<td></td>
</tr>
</tbody>
</table>

### Related Scenario

For similar scenarios using other databases, see:

- **Procedure** on page 625,  
- **Scenario: Removing and regenerating a MySQL table index** on page 2552.
**tEXistConnection**

Opens a connection to an eXist database in order that a transaction may be carried out.

**tEXistConnection Standard properties**

These properties are used to configure tEXistConnection running in the Standard Job framework.

The Standard tEXistConnection component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>URI</strong></td>
<td>URI of the database you want to connect to.</td>
</tr>
<tr>
<td><strong>Collection</strong></td>
<td>Enter the path to the collection of interest on the database server.</td>
</tr>
<tr>
<td><strong>Driver</strong></td>
<td>This field is automatically populated with the standard driver.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>User authentication information.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field,</td>
</tr>
<tr>
<td></td>
<td>and then in the pop-up dialog box enter the password between double quotes</td>
</tr>
<tr>
<td></td>
<td>and click OK to save the settings.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the job processing metadata at a Job level as</td>
</tr>
<tr>
<td></td>
<td>well as at each component level.</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage rule</strong></td>
<td>This component is more commonly used with other tEXist* components, especially</td>
</tr>
<tr>
<td></td>
<td>with the tEXistGet and tEXistPut components. If you set the connection</td>
</tr>
<tr>
<td></td>
<td>properties in the tEXistConnection component, you can reuse the connection</td>
</tr>
<tr>
<td></td>
<td>for other tEXist* components in the same Job.</td>
</tr>
<tr>
<td></td>
<td>eXist-db is an open source database management system built using XML</td>
</tr>
<tr>
<td></td>
<td>technology. It stores XML data according to the XML data model and features</td>
</tr>
<tr>
<td></td>
<td>efficient, index-based XQuery processing.</td>
</tr>
<tr>
<td></td>
<td>For further information about XQuery, see XQuery.</td>
</tr>
</tbody>
</table>

For further information about XQuery, see XQuery.
For further information about the XQuery update extension, see XQuery update extension.

<table>
<thead>
<tr>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

Related scenarios

For tEXistConnection related scenario, see tMysqlConnection on page 2475
tEXistDelete

Deletes specified resources from a remote eXist database.

tEXistDelete Standard properties

These properties are used to configure tEXistDelete running in the Standard Job framework.
The Standard tEXistDelete component belongs to the Databases family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Description</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection/Component List</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.</td>
</tr>
<tr>
<td>URI</td>
<td>URI of the database you want to connect to.</td>
</tr>
<tr>
<td>Collection</td>
<td>Enter the path to the collection of interest on the database server.</td>
</tr>
<tr>
<td>Driver</td>
<td>This field is automatically populated with the standard driver. Note: Users can enter a different driver, depending on their needs.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>User authentication information. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Target Type</td>
<td>Either Resource, Collection, or All.</td>
</tr>
<tr>
<td>Files</td>
<td>Click the plus button to add the lines you want to use as filters: Filemask: enter the filename or filemask using wildcharacters (*) or regular expressions.</td>
</tr>
</tbody>
</table>

Advanced settings

<table>
<thead>
<tr>
<th>Description</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the job processing metadata at a job level as well as at each component level.</td>
</tr>
</tbody>
</table>
### Global Variables

| Global Variables | NB_FILE: the number of files processed. This is an After variable and it returns an integer.  
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

### Usage

| Usage rule | This component is typically used as a single component sub-job but can also be used as an output or end object. eXist-db is an open source database management system built using XML technology. It stores XML data according to the XML data model and features efficient, index-based XQuery processing.  
For further information about XQuery, see XQuery.  
For further information about the XQuery update extension, see XQuery update extension. |

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |

### Related scenarios

No scenario is available for the Standard version of this component yet.
tEXistGet

Retrieves selected resources from a remote eXist database to a defined local directory.

**tEXistGet Standard properties**

These properties are used to configure tEXistGet running in the Standard Job framework.

The Standard tEXistGet component belongs to the Databases family.

The component in this framework is generally available.

### Basic settings

| Use an existing connection/Component List | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level. |
| URI | URI of the database you want to connect to. |
| Collection | Enter the path to the collection of interest on the database server. |
| Driver | This field is automatically populated with the standard driver. **Note:** Users can enter a different driver, depending on their needs. |
| Username and Password | User authentication information. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Local directory | Path to the file’s destination location. |
| Files | Click the plus button to add the lines you want to use as filters: **Filemask:** enter the filename or filemask using wildcharacters (*) or regular expressions |

### Advanced settings

| tStatCatcher Statistics | Select this check box to gather the job processing metadata at a job level as well as at each component level. |
Global Variables

**Global Variables**

- **NB_FILE**: the number of files processed. This is an After variable and it returns an integer.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**

This component is typically used as a single component sub-job but can also be used as an output or end object. eXist-db is an open source database management system built using XML technology. It stores XML data according to the XML data model and features efficient, index-based XQuery processing.

For further information about XQuery, see *XQuery*.

For further information about the XQuery update extension, see *XQuery update extension*.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Scenario: Retrieve resources from a remote eXist DB server

This is a single-component Job that retrieves data from a remote eXist DB server and download the data to a defined local directory.

This simple Job requires one component: **tEXistGet**.
Procedure

1. Drop the **tEXistGet** component from the **Palette** into the design workspace.
2. Double-click the **tEXistGet** component to open the Component view and define the properties in its **Basic settings** view.

3. Fill in the **URI** field with the URI of the eXist database you want to connect to.
   In this scenario, the URI is `xmlb:exist://192.168.0.165:8080/exist/xmlrpc`. Note that the URI used in this use case is for demonstration purposes only and is not an active address.

4. Fill in the **Collection** field with the path to the collection of interest on the database server, `/db/talend` in this scenario.

5. Fill in the **Driver** field with the driver for the XML database, `org.exist.xmldb.DatabaseImpl` in this scenario.

6. Fill in the **Username** and **Password** fields by typing in `admin` and `talend` respectively in this scenario.

7. Click the three-dot button next to the **Local directory** field to set a path for saving the XML file downloaded from the remote database server.
   In this scenario, set the path to your desktop, for example `C:/Documents and Settings/galano/Desktop/ExistGet`.

8. In the **Files** field, click the plus button to add a new line in the **Filemask** area, and fill it with a complete file name to retrieve data from a particular file on the server, or a filemask to retrieve data from a set of files. In this scenario, fill in `dictionary_en.xml`.

---

**tEXistGet**

**Basic settings**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td><code>xmlb:exist://192.168.0.165:8080/exist/xmlrpc</code></td>
</tr>
<tr>
<td>Collection</td>
<td><code>/db/talend</code></td>
</tr>
<tr>
<td>Driver</td>
<td><code>org.exist.xmldb.DatabaseImpl</code></td>
</tr>
<tr>
<td>Username</td>
<td><code>admin</code></td>
</tr>
<tr>
<td>Password</td>
<td><code>******</code></td>
</tr>
<tr>
<td>Local directory</td>
<td><code>C:/Documents and Settings/galano/Desktop/ExistGet</code></td>
</tr>
<tr>
<td>Filemask</td>
<td><code>dictionary_en.xml</code></td>
</tr>
</tbody>
</table>
9. Save your Job and press **F6** to execute it.

```xml
<!-- generated by ToXgene Version 1.1a in Wed Jun 23 12:16:00 EDT 2004 -->
<dictionary>
  <e id="E1">
    <hw>
      <ph>planks</ph>
      <pc>!g:kn+J|i6u</pc>
      <hw>garn</hw>
      <pc>2ft^OeH_Y</pc>
      <pos>adv.</pos>
    </hw>
    <et>
      <cr>E709</cr>
      <cr>E3</cr>
      <cr>E439</cr>
      <cr>E414</cr>
    </et>
  </e>
</dictionary>
```

The XML file `dictionary_en.xml` is retrieved and downloaded to the defined local directory.
tEXistList

Lists the resources stored on a remote eXist database.

**tEXistList Standard properties**

These properties are used to configure tEXistList running in the Standard Job framework.

The Standard tEXistList component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

| Use an existing connection/Component List | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.  

Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>URI of the database you want to connect to.</td>
</tr>
<tr>
<td>Collection</td>
<td>Enter the path to the collection of interest on the database server.</td>
</tr>
</tbody>
</table>
| Driver | This field is automatically populated with the standard driver.  

**Note:** Users can enter a different driver, depending on their needs. |
| Username and Password | Server authentication information.  

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Files | Click the plus button to add the lines you want to use as filters:  

**Filemask:** enter the filename or filemask using wildcharacters (*) or regular expressions. |
| Target Type | Either Resource, Collection or All contents: |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the job processing metadata at a job level as well as at each component level. |
# Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_FILE</strong>: the number of files iterated upon. This is an After variable, and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>CURRENT_FILE</strong>: the current file name. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>CURRENT_FILEPATH</strong>: the current file path. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <a href="https://help.talend.com">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is typically used along with a tEXistGet component to retrieve the files listed, for example.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>eXist-db is an open source database management system built using XML technology. It stores XML data according to the XML data model and features efficient, index-based XQuery processing.</td>
</tr>
<tr>
<td></td>
<td>For further information about XQuery, see <a href="https://help.talend.com">XQuery</a>. For further information about the XQuery update extension, see <a href="https://help.talend.com">XQuery update extension</a>.</td>
</tr>
</tbody>
</table>

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |

## Related scenario

For a related scenario, see [Listing and getting files/folders on an FTP directory](https://help.talend.com) on page 1232.
tEXistPut

Uploads specified files from a defined local directory to a remote eXist database.

tEXistPut Standard properties

These properties are used to configure tEXistPut running in the Standard Job framework.
The Standard tEXistPut component belongs to the Databases family.
The component in this framework is generally available.

Basic settings

| Use an existing connection/Component List | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level. |
| URI | URI of the database you want to connect to. |
| Collection | Enter a path to indicate where the resource is to be saved on the server. |
| Driver | This field is automatically populated with the standard driver. Note: Users can enter a different driver, depending on their needs. |
| Username and Password | User authentication information. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Local directory | Path to the source location of the file(s). |
| Files | Click the plus button to add the lines you want to use as filters: Filemask: enter the filename or filemask using wildcharacters (*) or regular expressions. |

Advanced settings

| tStatCatcher Statistics | Select this check box to gather the job processing metadata at a job level as well as at each component level. |
Global Variables

| Global Variables | NB_FILE: the number of files processed. This is an After variable and it returns an integer. ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component is typically used as a single component sub-job but can also be used as an output or end object. eXist-db is an open source database management system built using XML technology. It stores XML data according to the XML data model and features efficient, index-based XQuery processing. For further information about XQuery, see http://exist-db.org/exist/apps/doc/documentation.xml. For further information about the XQuery update extension, see http://exist-db.org/exist/apps/doc/update_ext.xml. |
| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |

Related scenarios

No scenario is available for the Standard version of this component yet.
tEXistXQuery

Queries XML files located on remote databases using local files containing XPath queries and outputs the results to an XML file stored locally.

**tEXistXQuery Standard properties**

These properties are used to configure tEXistXQuery running in the Standard Job framework.

The Standard tEXistXQuery component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection/Component List</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.</td>
</tr>
<tr>
<td>URI</td>
<td>URI of the database you want to connect to.</td>
</tr>
<tr>
<td>Collection</td>
<td>Enter the path to the XML file location on the database.</td>
</tr>
<tr>
<td>Driver</td>
<td>This field is automatically populated with the standard driver. Note: Users can enter a different driver, depending on their needs.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB server authentication information. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>XQuery Input File</td>
<td>Browse to the local file containing the query to be executed.</td>
</tr>
<tr>
<td>Local Output</td>
<td>Browse to the directory in which the query results should be saved.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the job processing metadata at a job level as well as at each component level.</td>
</tr>
</tbody>
</table>
## Global Variables

| Global Variables | NB_FILE: the number of files processed. This is an After variable and it returns an integer.  
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

## Usage

| Usage rule | This component is typically used as a single component Job but can also be used as part of a more complex Job.  
eXist-db is an open source database management system built using XML technology. It stores XML data according to the XML data model and features efficient, index-based XQuery processing.  
For further information about XQuery, see XQuery.  
For further information about the XQuery update extension, see XQuery update extension. |

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |

## Related scenarios

No scenario is available for the Standard version of this component yet.
tEXistXUpdate

Processes XML file records and updates the existing records on the database server.

**tEXistXUpdate Standard properties**

These properties are used to configure tEXistXUpdate running in the Standard Job framework.

The Standard tEXistXUpdate component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Use an existing connection/Component List</strong></th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, <strong>Component List</strong> presents only the connection components in the same Job level.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>URI</strong></td>
<td>URI of the database you want to connect to.</td>
</tr>
<tr>
<td><strong>Collection</strong></td>
<td>Enter the path to the collection and file of interest on the database server.</td>
</tr>
</tbody>
</table>
| **Driver** | This field is automatically populated with the standard driver.  
**Note:** Users can enter a different driver, depending on their needs. |
| **Username** and **Password** | DB server authentication information. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| **Update File** | Browse to the local file in the local directory to be used to update the records on the database. |

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the job processing metadata at a job level as well as at each component level. |

**Global Variables**

| **Global Variables** | **NB_FILE**: the number of files processed. This is an After variable and it returns an integer. |
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](https://help.talend.com).

### Usage

**Usage rule**

This component is typically used as a single component Job but can also be used as part of a more complex Job.

eXist-db is an open source database management system built using XML technology. It stores XML data according to the XML data model and features efficient, index-based XQuery processing.

For further information about XQuery, see [XQuery](https://help.talend.com).

For further information about the XQuery update extension, see [XQuery update extension](https://help.talend.com).

### Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### Related scenarios

No scenario is available for the Standard version of this component yet.
tExternalSortRow

Sorts input data based on one or several columns, by sort type and order, using an external sort application.

**tExternalSortRow Standard properties**

These properties are used to configure tExternalSortRow running in the Standard Job framework.

The Standard tExternalSortRow component belongs to the Processing family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either <strong>Built-in</strong> or stored remotely in the <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Sync columns</strong> to retrieve the schema from the previous component connected in the Job.</td>
</tr>
<tr>
<td>Built-in</td>
<td><strong>Built-in</strong>: The schema will be created and stored locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td>Repository</td>
<td><strong>Repository</strong>: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job flowcharts. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td>File Name</td>
<td><strong>File Name</strong>: Name or path to the file to be processed and/or the variable to be used. For further information about how to define and use a variable in a Job, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td>Field separator</td>
<td><strong>Field separator</strong>: Character, string or regular expression to separate fields.</td>
</tr>
</tbody>
</table>
### External command "sort" path
Enter the path to the external file containing the sorting algorithm to use.

### Criteria
Click the plus button to add as many lines as required for the sort to be complete. By default the first column defined in your schema is selected.

- **Schema column**: Select the column label from your schema, which the sort will be based on. Note that the order is essential as it determines the sorting priority.

- **Sort type**: Numerical and Alphabetical order are proposed. More sorting types to come.

- **Order**: Ascending or descending order.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Maximum memory</strong></td>
<td>Type in the size of physical memory you want to allocate to sort processing.</td>
</tr>
<tr>
<td><strong>Temporary directory</strong></td>
<td>Specify the temporary directory to process the sorting command.</td>
</tr>
<tr>
<td><strong>Set temporary input file directory</strong></td>
<td>Select the check box to activate the field in which you can specify the directory to handle your temporary input file.</td>
</tr>
<tr>
<td><strong>Add a dummy EOF line</strong></td>
<td>Select this check box when using the tAggregateSortRow component.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td><strong>NB_LINE</strong></td>
<td>the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).
Usage

| Usage rule | This component handles flow of data therefore it requires input and output, hence is defined as an intermediary step. |

Related scenario

For related use case, see tSortRow on page 3577.
**tExtractDelimitedFields**

Generates multiple columns from a delimited string column.

**tExtractDelimitedFields Standard properties**

These properties are used to configure tExtractDelimitedFields running in the Standard Job framework.

The Standard tExtractDelimitedFields component belongs to the Processing family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Field to split</th>
<th>Select an incoming field from the <strong>Field to split</strong> list to split.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignore NULL as the source data</td>
<td>Select this check box to ignore the Null value in the source data. Clear this check box to generate the Null records that correspond to the Null value in the source data.</td>
</tr>
<tr>
<td>Field separator</td>
<td>Enter character, string or regular expression to separate fields for the transferred data. <strong>Note:</strong> Since this component uses regex to split a filed and the regex syntax uses special characters as operators, make sure to precede the regex operator you use as a field separator by a double backslash. For example, you have to use <code>|</code> instead of `</td>
</tr>
<tr>
<td>Die on error</td>
<td>Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a <strong>Row &gt; Reject</strong> link.</td>
</tr>
</tbody>
</table>

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the
Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

Click Sync columns to retrieve the schema from the previous component connected in the Job.

<table>
<thead>
<tr>
<th>Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>Advanced settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advanced separator (for number)</strong></td>
<td>Select this check box to change the separator used for numbers. By default, the thousands separator is a comma (,) and the decimal separator is a period (.).</td>
</tr>
<tr>
<td><strong>Trim column</strong></td>
<td>Select this check box to remove leading and trailing whitespace from all columns.</td>
</tr>
<tr>
<td><strong>Check each row structure against schema</strong></td>
<td>Select this check box to check whether the total number of columns in each row is consistent with the schema. If not consistent, an error message will be displayed on the console.</td>
</tr>
<tr>
<td><strong>Validate date</strong></td>
<td>Select this check box to check the date format strictly against the input schema.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
<td></td>
</tr>
<tr>
<td><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
<td></td>
</tr>
<tr>
<td>For further information about variables, see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>
Usage

Usage rule

This component handles flow of data therefore it requires input and output components. It allows you to extract data from a delimited field, using a Row > Main link, and enables you to create a reject flow filtering data which type does not match the defined type.

Scenario: Extracting a delimited string column of a database table

This scenario describes a Job that writes data including a delimited string column into a MySQL database table and displays the data on the console, then extracts the delimited string column into multiple columns and displays the data after extraction on the console.

Adding and linking components

Procedure

1. Create a new Job and add the following components by typing their names in the design workspace or dropping them from the Palette: a tFixedFlowInput component, a tMysqlOutput component, a tMysqlInput component, a tExtractDelimitedFields component, two tLogRow components.

2. Link tFixedFlowInput to tMysqlOutput using a Row > Main connection.

3. Do the same to link tMysqlOutput to the first tLogRow, link tMysqlInput to tExtractDelimitedFields, link tExtractDelimitedFields to the second tLogRow.

4. Link tFixedFlowInput to tMysqlInput using a Trigger > On Subjob Ok connection.

Configuring the components

Populating data in a MySQL database table

Procedure

1. Double-click tFixedFlowInput to open its Basic settings view.
2. Click the [...] button next to Edit schema and in the pop-up window define the schema by adding three columns: Id of Integer type, and Name and DelimitedField of String type.

Click OK to close the schema editor and accept the propagation prompted by the pop-up dialog box.

3. In the Mode area, select Use Inline Content(delimited file). Then in the Content field displayed, enter the data to write to the database. This input data includes a delimited string column. In this example, the input data is as follows:

1;Adam;32,Component Team,Developer
2;Bill;28,Component Team,Tester
3;Chris;30,Doc Team,Writer
4;David;35,Doc Team,Leader
5;Eddie;33,QA Team,Tester

4. Double-click tMysqlOutput to open its Basic settings view.
5. Fill the **Host**, **Port**, **Database**, **Username**, **Password** fields with the MySQL database connection information.

6. Fill the **Table** field with the name of the table to be written. In this example, it is *employee*.

7. Select **Drop table if exists and create** from the **Action on table** list.

8. Double-click the first **tLogRow** to open its **Basic settings** view.

In the **Mode** area, select **Table (print values in cells of a table)** for better readability of the result.

**Extracting the delimited string column in the database table into multiple columns**

**Procedure**

1. Double-click **tMysqlInput** to open its **Basic settings** view.
2. Fill the **Host**, **Port**, **Database**, **Username**, **Password** fields with the MySQL database connection information.

3. Click the [...] button next to **Edit schema** and in the pop-up window define the schema of the **tMysqlInput** component same as the schema of the **tMysqlOutput** component.

4. In the **Table Name** field, enter the name of the table into which the data was written. In this example, it is *employee*.

5. Click the **Guess Query** button to fill the **Query** field with the SQL query statement to be executed on the specified table. In this example, it is as follows:

```sql
SELECT 'employee'.'Id', 'employee'.'Name', 'employee'.'DelimitedField'
FROM 'employee'
```

6. Double-click **tExtractDelimitedFields** to open its **Basic settings** view.
7. In the **Field to split** list, select the delimited string column to be extracted. In this example, it is **DelimitedField**.
   In the **Field separator**, enter the separator used to separate the fields in the delimited string column. In this example, it is `,`.
8. Click the ` [...] ` button next to **Edit schema** and in the pop-up window define the schema by adding five columns: **Id** of Integer type, and **Name**, **Age**, **Team**, **Title** of String type.
   In this example, the delimited string column **DelimitedField** is split into three columns **Age**, **Team** and **Title**, and the **Id** and **Name** columns are kept as well.

Click **OK** to close the schema editor and accept the propagation prompted by the pop-up dialog box.

9. Double-click the second **tLogRow** to open its **Basic settings** view.

In the **Mode** area, select **Table (print values in cells of a table)** for better readability of the result.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Execute the Job by pressing **F6** or clicking **Run** on the **Run** tab.
As shown above, the primitive input data and the data after extraction are displayed on the console, and the delimited string column `DelimitedField` is extracted into three columns `Age`, `Team`, and `Title`.

```
<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Age</th>
<th>Team</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Adam</td>
<td>32</td>
<td>Component Team</td>
<td>Developer</td>
</tr>
<tr>
<td>2</td>
<td>Bill</td>
<td>28</td>
<td>Component Team</td>
<td>Tester</td>
</tr>
<tr>
<td>3</td>
<td>Chris</td>
<td>30</td>
<td>Doc Team</td>
<td>Writer</td>
</tr>
<tr>
<td>4</td>
<td>David</td>
<td>35</td>
<td>Doc Team</td>
<td>Leader</td>
</tr>
<tr>
<td>5</td>
<td>Eddie</td>
<td>33</td>
<td>QA Team</td>
<td>Tester</td>
</tr>
</tbody>
</table>
```
tExtractJSONFields

Extracts the desired data from JSON fields based on the JSONPath or XPath query.

**tExtractJSONFields Standard properties**

These properties are used to configure tExtractJSONFields running in the Standard Job framework.

The Standard tExtractJSONFields component belongs to the Processing family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file where the properties are stored.</td>
</tr>
</tbody>
</table>

**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word *line* when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-In for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Read By**

Select a way of extracting JSON data in the file.

- **JsonPath**: Extracts JSON data based on the JSONPath query. With this option selected, you need to select a JSONPath API version from the **API version** drop-
down list. It is recommended to read data by JSONPath in order to gain better performance.

- **Xpath**: Extracts JSON data based on the XPath query.

<table>
<thead>
<tr>
<th>JSON field</th>
<th>List of the JSON fields to be extracted.</th>
</tr>
</thead>
</table>
| **Loop Jasonpath query** | Enter the path pointing to the node within the JSON field, on which the loop is based. 
Note if you have selected Xpath from the Read by drop-down list, the Loop Xpath query field is displayed instead. |
| **Mapping** | Complete this table to map the columns defined in the schema to the corresponding JSON nodes. 
- **Column**: The Column cells are automatically filled with the defined schema column names. 
- **Json query/JSONPath query**: Specify the JSONPath node that holds the desired data. For more information about JSONPath expressions, see [http://goessner.net/articles/JsonPath/](http://goessner.net/articles/JsonPath/). 
  This column is available only when JsonPath is selected from the Read By list. 
- **XPath query**: Specify the XPath node that holds the desired data. 
  This column is available only when Xpath is selected from the Read By list. 
- **Get Nodes**: Select this check box to extract the JSON data of all the nodes or select the check box next to a specific node to extract the data of that node. 
  This column is available only when Xpath is selected from the Read By list. 
- **Is Array**: select this check box when the JSON field to be extracted is an array instead of an object. 
  This column is available only when Xpath is selected from the Read By list. |
| **Die on error** | Select the check box to stop the execution of the Job when an error occurs. 
Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row > Reject link. |

**Advanced settings**

| Encoding | Select the encoding from the list or select Custom and define it manually. This field is compulsory for database data handling. |
### tStatCatcher Statistics

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is selected.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage rule</strong></td>
<td>This component is an intermediate component. It needs an input and an output components.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Limitation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Limitation</strong></td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the <a href="https://help.talend.com">Install</a> button on the <strong>Component</strong> tab view. You can also find out and add all missing JARs easily on the <strong>Modules</strong> tab in the <strong>Integration</strong> perspective of your studio. You can find more details about how to install external modules in <a href="https://help.talend.com">Talend Help Center</a>.</td>
</tr>
</tbody>
</table>

### Scenario 1: Retrieving error messages while extracting data from JSON fields

In this scenario, `tWriteJSONField` wraps the incoming data into JSON fields, data of which is then extracted by `tExtractJSONFields`. Meanwhile, the error messages generated due to extraction failure, which include the concerned JSON fields and errors, are retrieved via a **Row > Reject** link.

### Linking the components

**Procedure**

1. Drop the following components from the **Palette** onto the design workspace: `tFixedFlowInput`, `tWriteJSONField`, `tExtractJSONFields`, and `tLogRow (X2)`. The two `tLogRow` components are renamed as `data_extracted` and `reject_info`.
2. Link `tFixedFlowInput` and `tWriteJSONField` using a **Row > Main** connection.
3. Link `tWriteJSONField` and `tExtractJSONFields` using a **Row > Main** connection.
4. Link `tExtractJSONFields` and `data_extracted` using a **Row > Main** connection.
5. Link `tExtractJSONFields` and `reject_info` using a **Row > Reject** connection.
Configuring the components

Setting up the tFixedFlowInput

Procedure

1. Double-click tFixedFlowInput to display its Basic settings view.

2. Click Edit schema to open the schema editor.
Click the [+] button to add three columns, namely *firstname*, *lastname* and *dept*, with the type of string.

Click **OK** to close the editor.

3. Select **Use Inline Content** and enter the data below in the **Content** box:

<table>
<thead>
<tr>
<th>Andrew; Wallace; Doc</th>
</tr>
</thead>
<tbody>
<tr>
<td>John; Smith; R&amp;D</td>
</tr>
<tr>
<td>Christian; Dior; Sales</td>
</tr>
</tbody>
</table>

**Setting up the tWriteJSONField**

**Procedure**

1. Click **tWriteJSONField** to display its **Basic settings** view.

![Basic settings](image)

2. Click **Configure JSON Tree** to open the XML tree editor.

![XML tree editor](image)

   The schema of **tFixedFlowInput** appears in the **Linker source** panel.

3. In the **Linker target** panel, click the default **rootTag** and type in **staff**, which is the root node of the JSON field to be generated.

4. Right-click **staff** and select **Add Sub-element** from the context menu.

5. In the pop-up box, enter the sub-node name, namely **firstname**.
Repeat the steps to add two more sub-nodes, namely *lastname* and *dept*.

6. Right-click *firstname* and select **Set As Loop Element** from the context menu.
7. Drop *firstname* from the **Linker source** panel to its counterpart in the **Linker target** panel. In the pop-up dialog box, select **Add linker to target node**.

Click **OK** to close the dialog box.

8. Repeat the steps to link the two other items.
   Click **OK** to close the XML tree editor.
9. Click **Edit schema** to open the schema editor.

10. Click the [+] button in the right panel to add one column, namely *staff*, which will hold the JSON data generated.
    Click **OK** to close the editor.
Setting up the tExtractJSONFields

Procedure

1. Double-click tExtractJSONFields to display its Basic settings view.

![tExtractJSONFields](image1)

2. Click Edit schema to open the schema editor.

![Schema editor](image2)

3. Click the [+] button in the right panel to add three columns, namely firstname, lastname and dept, which will hold the data of their counterpart nodes in the JSON field staff.

   Click OK to close the editor.

4. In the pop-up Propagate box, click Yes to propagate the schema to the subsequent components.
5. In the **Loop XPath query** field, enter "/staff", which is the root node of the JSON data.

6. In the **Mapping** area, type in the node name of the JSON data under the **XPath query** part. The data of those nodes will be extracted and passed to their counterpart columns defined in the output schema.

7. Specifically, define the XPath query "firstname" for the column firstname, 'lastname" for the column lastname, and "" for the column dept. Note that "" is not a valid XPath query and will lead to execution errors.

---

### Setting up the tLogRow components

**Procedure**

1. Double-click **data_extracted** to display its **Basic settings** view.

   ![data_extracted](image)

2. Select **Table (print values in cells of a table)** for a better display of the results.

3. Perform the same setup on the other **tLogRow** component, namely **reject_info**.

### Executing the Job

**Procedure**

1. Press **Ctrl + S** to save the Job.

2. Click **F6** to execute the Job.

```
<table>
<thead>
<tr>
<th>firstname</th>
<th>lastname</th>
<th>dept</th>
<th>errorJSONField</th>
<th>errorCode</th>
<th>errorMessage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew</td>
<td>Wallace</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>Invalid XPath expression: Unexpected</td>
</tr>
<tr>
<td>John</td>
<td>Smith</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>Invalid XPath expression: Unexpected</td>
</tr>
<tr>
<td>Christian</td>
<td>Door</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>Invalid XPath expression: Unexpected</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>data_extracted</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstname</td>
</tr>
<tr>
<td>-----------</td>
</tr>
</tbody>
</table>
```

As shown above, the reject row offers such details as the data extracted, the JSON fields whose data is not extracted and the cause of the extraction failure.

---

### Scenario 2: Collecting data from your favorite online social network

In this scenario, **tFileInputJSON** retrieves the **friends** node from a JSON file that contains the data of a Facebook user and **tExtractJSONFields** extracts the data from the **friends** node for flat data output.
Linking the components

Procedure
1. Drop the following components from the Palette onto the design workspace: tFileInputJSON, tExtractJSONFields and tLogRow.
2. Link tFileInputJSON and tExtractJSONFields using a Row > Main connection.
3. Link tExtractJSONFields and tLogRow using a Row > Main connection.

Configuring the components

Procedure
1. Double-click tFileInputJSON to display its Basic settings view.
2. Click Edit schema to open the schema editor.
Click the [+] button to add one column, namely friends, of the String type.
Click OK to close the editor.

3. Click the [...] button to browse for the JSON file, facebook.json in this case:

```json
{ "user": { "id": "9999912398", "name": "Kelly Clarkson", "friends": [ { "name": "Tom Cruise", "id": "55555555555555", "likes": { "data": [ { "category": "Movie", "name": "The Shawshank Redemption", "id": "103636093053996", "created_time": "2012-11-20T15:52:07+0000" }, { "category": "Community", "name": "Positiveretribution", "id": "471389562899413", "created_time": "2012-12-16T21:13:26+0000" } ] } }, { "name": "Tom Hanks", "id": "88888888888888", "likes": { "data": [ { "category": "Journalist", "name": "Janelle Wang", "id": "136009823148851", "created_time": "2013-01-01T08:22:17+0000" }, { "category": "Tv show", "name": "Now With Alex Wagner", "id": "305948749433410", "created_time": "2012-11-20T06:14:10+0000" } ] } ] } }
```

4. Clear the Read by XPath check box.
   In the Mapping table, enter the JSONPath query ".user.friends[*]" next to the friends column, retrieving the entire friends node from the source file.

5. Double-click tExtractJSONFields to display its Basic settings view.
6. Click **Edit schema** to open the schema editor.

7. Click the [+] button in the right panel to add five columns, namely *id*, *name*, *like_id*, *like_name* and *like_category*, which will hold the data of relevant nodes in the JSON field *friends*. Click **OK** to close the editor.

8. In the pop-up **Propagate** box, click **Yes** to propagate the schema to the subsequent components.

9. In the **Loop XPath query** field, enter “/likes/data”.
10. In the **Mapping** area, type in the queries of the JSON nodes in the **XPath query** column. The data of those nodes will be extracted and passed to their counterpart columns defined in the output schema.

11. Specifically, define the XPath query "../../id" (querying the "friends/id" node) for the column **id**, "../../name" (querying the "friends/name" node) for the column **name**, "id" for the column **like_id**, "name" for the column **like_name**, and "category" for the column **like_category**.

12. Double-click **tLogRow** to display its **Basic settings** view.

13. Select **Table (print values in cells of a table)** for a better display of the results.

**Executing the Job**

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Click **F6** to execute the Job.

As shown above, the friends data of the Facebook user Kelly Clarkson is extracted correctly.
tExtractPositionalFields

Extracts data and generates multiple columns from a formatted string using positional fields. tExtractPositionalFields generates multiple columns from one column using positional fields.

**tExtractPositionalFields Standard properties**

These properties are used to configure tExtractPositionalFields running in the Standard Job framework.

The Standard tExtractPositionalFields component belongs to the Processing family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Field</strong></td>
<td>Select an incoming field from the Field list to extract.</td>
</tr>
<tr>
<td><strong>Ignore NULL as the source data</strong></td>
<td>Select this check box to ignore the Null value in the source data.</td>
</tr>
<tr>
<td></td>
<td>Clear this check box to generate the Null records that correspond to the Null value in the source data.</td>
</tr>
<tr>
<td><strong>Customize</strong></td>
<td>Select this check box to customize the data format of the positional file and define the table columns: Column: Select the column you want to customize. Size: Enter the column size. Padding char: Type in between inverted commas the padding character used, in order for it to be removed from the field. A space by default. Alignment: Select the appropriate alignment parameter.</td>
</tr>
<tr>
<td><strong>Pattern</strong></td>
<td>Enter the pattern to use as basis for the extraction.</td>
</tr>
<tr>
<td></td>
<td>A pattern is length values separated by commas, interpreted as a string between quotes. Make sure the values entered in this fields are consistent with the schema defined.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row &gt; Reject link.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
</tbody>
</table>
• **View schema**: choose this option to view the schema only.

• **Change to built-in property**: choose this option to change the schema to Built-in for local changes.

• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

### Advanced settings

<table>
<thead>
<tr>
<th>Advanced separator (for number)</th>
<th>Select this check box to change the separator used for numbers. By default, the thousands separator is a comma (,) and the decimal separator is a period (.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trim Column</td>
<td>Select this check box to remove leading and trailing whitespace from all columns.</td>
</tr>
<tr>
<td>Check each row structure against schema</td>
<td>Select this check box to check whether the total number of columns in each row is consistent with the schema. If not consistent, an error message will be displayed on the console.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>

The **NB_LINE** variable is not available to the Map/Reduce version.
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

| Usage rule | This component handles flow of data therefore it requires input and output components. It allows you to extract data from a delimited field, using a Row > Main link, and enables you to create a reject flow filtering data which type does not match the defined type. |

Related scenario

For a related scenario, see Scenario: Extracting name, domain and TLD from e-mail addresses on page 963.
tExtractRegexFields

Extracts data and generates multiple columns from a formatted string using regex matching.

**tExtractRegexFields Standard properties**

These properties are used to configure tExtractRegexFields running in the Standard Job framework.

The Standard tExtractRegexFields component belongs to the Data Quality and the Processing families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Field to split</th>
<th>Select an incoming field from the Field to split list to split.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regex</td>
<td>Enter a regular expression according to the programming language you are using.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td></td>
<td>Click Sync columns to retrieve the schema from the previous component connected in the Job.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning:</strong> Make sure that the output schema does not contain any column with the same name as the input column to be split. Otherwise, the regular expression will not work as expected.</td>
</tr>
<tr>
<td>Built-In</td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

Advanced settings

| **Die on error** | Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row > Reject link. |
| **Check each row structure against schema** | Select this check box to check whether the total number of columns in each row is consistent with the schema. If not consistent, an error message will be displayed on the console. |
| **tStatCatcher Statistics** | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

Global Variables

| **Global Variables** | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| **Usage rule** | This component handles flow of data therefore it requires input and output components. It allows you to extract data from a delimited field, using a Row > Main link, and enables you to create a reject flow filtering data which type does not match the defined type. |

Scenario: Extracting name, domain and TLD from e-mail addresses

This scenario describes a three-component Job where tExtractRegexFields is used to specify a regular expression that corresponds to one column in the input data, email. The tExtractRegexFields component handles flow of data therefore it requires input and output components. It allows you to extract data from a delimited field, using a Row > Main link, and enables you to create a reject flow filtering data which type does not match the defined type.
The **tExtractRegexFields** component is used to perform the actual regular expression matching. This regular expression includes field identifiers for user name, domain name and Top-Level Domain (TLD) name portions in each e-mail address. If the given e-mail address is valid, the name, domain and TLD are extracted and displayed on the console in three separate columns. Data in the other two input columns, *id* and *age* is extracted and routed to destination as well.

### Setting up the Job

**Procedure**

1. Drop the following components from the Palette onto the design workspace: **tFileInputDelimited**, **tExtractRegexFields**, and **tLogRow**.

2. Connect **tFileInputDelimited** to **tExtractRegexFields** using a Row > Main link, and do the same to connect **tExtractRegexFields** to **tLogRow**.

### Configuring the components

**Procedure**

1. Double-click the **tFileInputDelimited** component to open its Basic settings view in the Component tab.

2. Click the [...] button next to the File name/Stream field to browse to the file where you want to extract information from.

   The input file used in this scenario is called *test4*. It is a text file that holds three columns: *id*, *email*, and *age*.

```plaintext
id;email;age
1;anna@yahoo.net;24
2;diana@sohu.com;31
3;fiona@gmail.org;20
```
For more information, see `tFileInputDelimited` on page 1010.

3. Click **Edit schema** to define the data structure of this input file.

4. Double-click the **tExtractRegexFields** component to open its **Basic settings** view.

5. Select the column to split from the Field to split list: `email` in this scenario.

6. Enter the regular expression you want to use to perform data matching in the Regex panel. In this scenario, the regular expression `"([a-z]*@[a-z]*\.[a-z]*)"` is used to match the three parts of an email address: user name, domain name and TLD name.


7. Click **Edit schema** to open the **[Schema of tExtractRegexFields]** dialog box, and click the plus button to add five columns for the output schema.

   In this scenario, we want to split the input `email` column into three columns in the output flow, `name`, `domain`, and `tld`. The two other input columns will be extracted as they are.

8. Double-click the **tLogRow** component to open its **Component** view.

9. In the Mode area, select **Table (print values in cells of a table)**.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.

2. Execute the Job by pressing **F6** or clicking **Run** on the **Run** tab.
Results

The **tExtractRegexFields** component matches all given e-mail addresses with the defined regular expression and extracts the name, domain, and TLD names and displays them on the console in three separate columns. The two other columns, *id* and *age*, are extracted as they are.
**tExtractXMLField**

Reads the XML structured data from an XML field and sends the data as defined in the schema to the following component.

**tExtractXMLField Standard properties**

These properties are used to configure tExtractXMLField running in the Standard Job framework. The Standard tExtractXMLField component belongs to the Processing and the XML families. The component in this framework is generally available.

**Basic settings**

| Property type | Either Built-In or Repository. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
- View schema: choose this option to view the schema only.  
- Change to built-in property: choose this option to change the schema to Built-in for local changes.  
- Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
| Built-In: No property data stored centrally. |
| Repository: Select the repository file where the properties are stored. When this file is selected, the fields that follow are pre-filled in using fetched data. |

**Schema type and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

| Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide. |
| Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. |

**XML field**

Name of the XML field to be processed.
## Loop XPath query

Node of the XML tree, which the loop is based on.

### Related topic: see *Talend Studio User Guide*.

## Mapping

- **Column**: reflects the schema as defined by the Schema type field.
- **XPath Query**: Enter the fields to be extracted from the structured input.
- **Get nodes**: Select this check box to recuperate the XML content of all current nodes specified in the XPath query list or select the check box next to specific XML nodes to recuperate only the content of the selected nodes.

## Limit

Maximum number of rows to be processed. If Limit is 0, no rows are read or processed.

## Die on error

Select the check box to stop the execution of the Job when an error occurs.

Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row > Reject link.

## Advanced settings

- **Ignore the namespaces**: Select this check box to ignore namespaces when reading and extracting the XML data.
- **tStatCatcher Statistics**: Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

## Global Variables

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.
- **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*. 
Usage

| Usage rule | This component is an intermediate component. It needs an input and an output components. |

Scenario 1: Extracting XML data from a field in a database table

This three-component scenario allows to read the XML structure included in the fields of a database table and then extracts the data.

Procedure

1. Drop the following components from the Palette onto the design workspace: tMysqlInput, tExtractXMLField, and tFileOutputDelimited. Connect the three components using Main links.

2. Double-click tMysqlInput to display its Basic settings view and define its properties.

3. If you have already stored the input schema in the Repository tree view, select Repository first from the Property Type list and then from the Schema list to display the [Repository Content] dialog box where you can select the relevant metadata.

   For more information about storing schema metadata in the Repository tree view, see Talend Studio User Guide.

   If you have not stored the input schema locally, select Built-in in the Property Type and Schema fields and enter the database connection and the data structure information manually. For more information about tMysqlInput properties, see tMysqlInput on page 2488.
4. In the **Table Name** field, enter the name of the table holding the XML data, *customerdetails* in this example.

   Click **Guess Query** to display the query corresponding to your schema.

5. Double-click **tExtractXMLField** to display its **Basic settings** view and define its properties.

6. Click **Sync columns** to retrieve the schema from the preceding component. You can click the three-dot button next to **Edit schema** to view/modify the schema.

   The **Column** field in the **Mapping** table will be automatically populated with the defined schema.

7. In the **Xml field** list, select the column from which you want to extract the XML data. In this example, the field holding the XML data is called *CustomerDetails*.

   In the **Loop XPath query** field, enter the node of the XML tree on which to loop to retrieve data.

   In the **Xpath query** column, enter between inverted commas the node of the XML field holding the data you want to extract, *CustomerName* in this example.

8. Double-click **tFileOutputDelimited** to display its **Basic settings** view and define its properties.

9. In the **File Name** field, define or browse to the path of the output file you want to write the extracted data in.

   Click **Sync columns** to retrieve the schema from the preceding component. If needed, click the three-dot button next to **Edit schema** to view the schema.

10. Save your Job and click **F6** to execute it.
**Results**

1. Griffith Paving and Sealcoating
2. Bill’s Dive Shop
3. Childress Child Day Care
4. Facelift Kitchen and Bath
5. Terrini & Son Auto and Truck
6. Kermit the Pet Shop
7. Tub's Furniture Store
8. Toggle & Myerson Ltd
9. Childress Child Day Care
10. Elle Hypnosis and Therapy Cent
11. Lennox Air Pollution Control

The *tExtractXMLField* read and extracted the clients names under the node *CustomerName* of the *CustomerDetails* field of the defined database table.

**Scenario 2: Extracting correct and erroneous data from an XML field in a delimited file**

This scenario describes a four-component Job that reads an XML structure from a delimited file, outputs the main data and rejects the erroneous data.

**Procedure**

1. Drop the following components from the *Palette* to the design workspace: *tFileInputDelimited*, *tExtractXMLField*, *tFileOutputDelimited* and *tLogRow*.
   
   Connect the first three components using *Row Main* links.
   
   Connect *tExtractXMLField* to *tLogRow* using a *Row Reject* link.

2. Double-click *tFileInputDelimited* to open its *Basic settings* view and define the component properties.
3. Select **Built-in** in the **Schema** list and fill in the file metadata manually in the corresponding fields.

   Click the three-dot button next to **Edit schema** to display a dialog box where you can define the structure of your data.

   Click the plus button to add as many columns as needed to your data structure. In this example, we have one column in the schema: `xmlStr`.

   Click **OK** to validate your changes and close the dialog box.

   **Note:**

   If you have already stored the schema in the **Metadata** folder under **File delimited**, select **Repository** from the **Schema** list and click the three-dot button next to the field to display the [Repository Content] dialog box where you can select the relevant schema from the list. Click **Ok** to close the dialog box and have the fields automatically filled in with the schema metadata.

   For more information about storing schema metadata in the Repository tree view, see *Talend Studio User Guide*.

4. In the **File Name** field, click the three-dot button and browse to the input delimited file you want to process, `CustomerDetails_Error` in this example.

   This delimited file holds a number of simple XML lines separated by double carriage return.

   Set the row and field separators used in the input file in the corresponding fields, double carriage return for the first and nothing for the second in this example.

   If needed, set **Header**, **Footer** and **Limit**. None is used in this example.

5. In the design workspace, double-click **tExtractXMLField** to display its **Basic settings** view and define the component properties.
6. Click **Sync columns** to retrieve the schema from the preceding component. You can click the three-dot button next to **Edit schema** to view/modify the schema.

   The **Column** field in the **Mapping** table will be automatically populated with the defined schema.

7. In the **Xml field** list, select the column from which you want to extract the XML data. In this example, the file holding the XML data is called **xmlStr**.

   In the **Loop XPath query** field, enter the node of the XML tree on which to loop to retrieve data.

8. In the design workspace, double-click **tFileOutputDelimited** to open its **Basic settings** view and display the component properties.

9. In the **File Name** field, define or browse to the output file you want to write the correct data in, **CustomerNames_right.csv** in this example.

   Click **Sync columns** to retrieve the schema of the preceding component. You can click the three-dot button next to **Edit schema** to view/modify the schema.

10. In the design workspace, double-click **tLogRow** to display its **Basic settings** view and define the component properties.

    Click **Sync Columns** to retrieve the schema of the preceding component. For more information on this component, see **tLogRow** on page 1988.

11. Save your Job and press **F6** to execute it.
Results


[statistics] connecting to socket on port 3845
[statistics] connected

<CustomerDetails>
  <CustomerName>Childress Child Day Care</CustomerName>
</CustomerDetails>

Error on line 3 of document : The element type "CustomerName" must be terminated by the matching end-tag "</CustomerName>". Nested exception: The element type "CustomerName" must be terminated by the matching end-tag "</CustomerName>". - Line: 8

Job A ended at 14:22 05/11/2009. [exit code=0]

tExtractXMLField reads and extracts in the output delimited file, CustomerNames_right, the client information for which the XML structure is correct, and displays as well erroneous data on the console of the Run view.
tFileArchive

Creates a new zip, gzip, or tar.gz archive file from one or more files or folders.
The archive file can be compressed using different compression method.

**tFileArchive Standard properties**

These properties are used to configure tFileArchive running in the Standard Job framework.
The Standard tFileArchive component belongs to the File family.
The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directory</strong></td>
<td>Specify the directory that contains the files to be added to the archive file. This field is available when zip or tar.gz is selected from the Archive format list.</td>
</tr>
<tr>
<td><strong>Subdirectories</strong></td>
<td>Select this check box if you want to add the files in the subdirectories to the archive file. This field is available only when zip is selected from the Archive format list.</td>
</tr>
<tr>
<td><strong>Source File</strong></td>
<td>Specify the path to the file that you want to add to the archive file. This field is available only when gzip is selected from the Archive format list.</td>
</tr>
<tr>
<td><strong>Archive file</strong></td>
<td>Specify the path to the archive file to be created.</td>
</tr>
<tr>
<td><strong>Create directory if does not exist</strong></td>
<td>Select this check box to create the destination folder if it does not exist.</td>
</tr>
<tr>
<td><strong>Archive format</strong></td>
<td>Select an archive file format from the list: zip, gzip, or tar.gz.</td>
</tr>
<tr>
<td><strong>Compress level</strong></td>
<td>Select the compression level you want to apply.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Best</strong>: the compression quality will be optimum, but the compression time will be long.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Normal</strong>: the compression quality and time will be average.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Fast (no compression)</strong>: the compression will be fast, but the quality will be lower.</td>
</tr>
<tr>
<td><strong>All files</strong></td>
<td>Select this check box if all files in the specified directory will be added to the archive file. Clear it to specify the file(s) you want to add to the archive file in the Files table.</td>
</tr>
<tr>
<td></td>
<td><strong>Filemask</strong>: type in the file name or the file mask using a special character or a regular expression.</td>
</tr>
<tr>
<td>Setting</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>FileArchive</strong></td>
<td>This check box is available when <code>zip</code> or <code>tar.gz</code> is selected from the Archive format list.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select an encoding type from the list or select CUSTOM and define it manually. This field is compulsory for DB data handling. This list is available when <code>zip</code> is selected from the Archive format list.</td>
</tr>
<tr>
<td><strong>Overwrite Existing Archive</strong></td>
<td>This check box is selected by default. This allows you to save an archive by replacing the existing one. But if you clear the check box, an error is reported, the replacement fails and the new archive cannot be saved. Note: When the replacement fails, the Job runs.</td>
</tr>
<tr>
<td><strong>Encrypt files</strong></td>
<td>Select this check box if you want the archive file to be password protected. Encrypt method: select an encrypt method from the list, Java Encrypt, Zip4j AES, or Zip4j STANDARD. AES Key Strength: select a key strength for the Zip4j AES method, either AES 128 or AES 256. Enter Password: enter the encryption password. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. This check box is available only when <code>zip</code> is selected from the Archive format list. With this check box selected, the compressed archive file can be decompressed only by the tFileUnarchive component and not by a common archiver. For more information about tFileUnarchive, see tFileUnarchive on page 1169.</td>
</tr>
<tr>
<td><strong>ZIP64 mode</strong></td>
<td>This option allows for archives with the .zip64 extension to be created, with three modes available: • ASNEEDED: archives with the .zip64 extension will be automatically created based on the file size. • ALWAYS: archives with the .zip64 extension will be created, no matter what size the file may be. • NEVER: no archives with the .zip64 extension will be created, no matter what size the file may be. Note that if the file size or the total size of the archive exceeds 4GB or there are more than 65536 files inside the archive, you need to set the mode to ALWAYS.</td>
</tr>
<tr>
<td><strong>Advanced settings</strong></td>
<td>Use sync flush: Select this check box to flush the compressor before flushing the output stream. Clear this check box to flush only the output stream.</td>
</tr>
</tbody>
</table>

976
This check box is available when gzip or tar.gz is selected from the Archive format list.

**tStatCatcher Statistics**

Select this check box to gather the processing metadata at the Job level as well as at each component level.

**Global Variables**

**Global Variables**

- **ARCHIVE_FILEPATH**: the path to the archive file. This is an After variable and it returns a string.
- **ARCHIVE_FILENAME**: the name of the archive file. This is an After variable and it returns a string.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

**Usage**

**Usage rule**

This component must be used as a standalone component.

**Connections**

Outgoing links (from this component to another):
- **Row**: Main; Reject; Iterate.
- **Trigger**: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error.

Incoming links (from one component to this one):
- **Row**: Main; Reject; Iterate.
- **Trigger**: Run if; On Subjob Ok; On Subjob Error; On component Ok; On Component Error; Synchronize; Parallelize.

For further information regarding connections, see Talend Studio User Guide.

**Scenario: Zip files using a tFileArchive**

This scenario creates a Job with a unique component. It aims at zipping files and recording them in the selected directory.
Procedure

1. Drop the **tFileArchive** component from the **Palette** onto the workspace.
2. Double-click it to display its **Component** view.

3. In the **Directory** field, click the [...] button, browse your directory and select the directory or the file you want to compress.
4. Select the **Subdirectories** check box if you want to include the subfolders and their files in the archive.
5. Then, set the **Archive file** field, by filling the destination path and the name of your archive file.
6. Select the **Create directory if not exists** check box if you do not have a destination directory yet and you want to create it.
7. In the **Compress level** list, select the compression level you want to apply to your archive. In this example, we use the **normal** level.
8. Clear the **All Files** check box if you only want to zip specific files.
9. Add a row in the table by clicking the [+\+] button and click the name which appears. Between two star symbols (ie. *RG*), type part of the name of the file that you want to compress.

10. Press **F6** to execute your Job.

**Results**

The **tFileArchive** has compressed the selected file(s) and created the folder in the selected directory.
tFileCompare

Compares two files and provides comparison data based on a read-only schema.

**tFileCompare Standard properties**

These properties are used to configure tFileCompare running in the Standard Job framework.
The Standard tFileCompare component belongs to the File family.
The component in this framework is generally available.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description, it defines the number of fields to be processed and passed on to the next component.
The schema of this component is read-only. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File to compare</strong></td>
<td>Filepath to the file to be checked.</td>
</tr>
<tr>
<td><strong>Reference file</strong></td>
<td>Filepath to the file, the comparison is based on.</td>
</tr>
<tr>
<td><strong>If differences are detected, display</strong> and <strong>If no difference detected, display</strong></td>
<td>Type in a message to be displayed in the Run console based on the result of the comparison.</td>
</tr>
<tr>
<td><strong>Print to console</strong></td>
<td>Select this check box to display the message.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Encoding</strong></th>
<th>Select the encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>DIFFERENCE</strong>: the result of the comparison. This is a Flow variable and it returns a boolean.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
</tbody>
</table>
Usage

Usage rule

This component can be used as standalone component but it is usually linked to an output component to gather the log data.

Connections

Outgoing links (from this component to another):
- **Row**: Main.
- **Trigger**: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error.

Incoming links (from one component to this one):
- **Row**: Main; Reject; Iterate.
- **Trigger**: Run if; On Subjob Ok; On Subjob Error; On component Ok; On Component Error; Synchronize; Parallelize.

For further information regarding connections, see *Talend Studio User Guide*.

Scenario: Comparing unzipped files

This scenario describes a Job unarchiving a file and comparing it to a reference file to make sure it did not change. The output of the comparison is stored into a delimited file and a message displays in the console.

![Diagram of component connections]

Procedure

1. Drag and drop the following components: tFileUnarchive, tFileCompare, and tFileOutputDelimited.
2. Link the tFileUnarchive to the tFileCompare with Iterate connection.
3. Connect the tFileCompare to the output component, using a Main row link.
4. In the tFileUnarchive component Basic settings, fill in the path to the archive to unzip.
5. In the Extraction Directory field, fill in the destination folder for the unarchived file.
6. In the tFileCompare Basic settings, set the File to compare. Press Ctrl+Space bar to display the list of global variables. Select $globals[tFileUnarchive_1][CURRENT_FILEPATH] or "((String)glob..."
1. Use the alMap.get("tFileUnarchive_1_CURRENT_FILEPATH")* method to fetch the file path from the tFileUnarchive component.

7. Set the **Reference file** to base the comparison on it.

8. In the messages fields, set the messages you want to see if the files differ or if the files are identical, for example: "[job ' + JobName + ' Files differ]."

9. Select the **Print to Console** check box, for the message defined to display at the end of the execution.

10. The schema is read-only and contains standard information data. Click **Edit schema** to have a look to it.

11. Then set the output component as usual with semi-colon as data separators.

12. Save your Job and press **F6** to run it.

```
Starting job CompareFiles at 14:11 19/06/2007.
[job CompareFiles] Files differ
Job CompareFiles ended at 14:11 19/06/2007. [exit code=0]
```

The message set is displayed to the console and the output shows the schema information data.
**tFileCopy**

Copies a source file or folder into a target directory.

**tFileCopy Standard properties**

These properties are used to configure tFileCopy running in the Standard Job framework.

The Standard tFileCopy component belongs to the File family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File Name</strong></td>
<td>Specify the path to the file to be copied. This field does not appear when the <strong>Copy a directory</strong> check box is selected.</td>
</tr>
<tr>
<td></td>
<td><strong>Copy a directory</strong> Select this check box to copy a directory including all subdirectories and files in it.</td>
</tr>
<tr>
<td><strong>Source directory</strong></td>
<td>Specify the source directory to copy. This field appears only when the <strong>Copy a directory</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Destination directory</strong></td>
<td>Specify the directory to copy the source file or directory to.</td>
</tr>
<tr>
<td><strong>Rename</strong></td>
<td>Select this check box if you want to rename the file copied to the destination. This field does not appear when the <strong>Copy a directory</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Destination filename</strong></td>
<td>Specify a new name for the file to be copied. This field appears only when the <strong>Rename</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Remove source file</strong></td>
<td>Select this check box to remove the source file after it is copied to the destination directory. This field does not appear when the <strong>Copy a directory</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Replace existing file</strong></td>
<td>Select this check box to overwrite any existing file with the newly copied file. This field does not appear when the <strong>Copy a directory</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Create the directory if it doesn’t exist</strong></td>
<td>Select this check box to create the specified destination directory if it does not exist. This field does not appear when the <strong>Copy a directory</strong> check box is selected.</td>
</tr>
</tbody>
</table>
### Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DESTINATION_FILENAME</strong>: the destination file name. This is an After variable and it returns a string.</td>
<td></td>
</tr>
<tr>
<td><strong>DESTINATION_FILEPATH</strong>: the destination file path. This is an After variable and it returns a string.</td>
<td></td>
</tr>
<tr>
<td><strong>SOURCE_DIRECTORY</strong>: the source directory. This is an After variable and it returns a string.</td>
<td></td>
</tr>
<tr>
<td><strong>DESTINATION_DIRECTORY</strong>: the destination directory. This is an After variable and it returns a string.</td>
<td></td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
<td></td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as a standalone component.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Connections</th>
<th>Outgoing links (from this component to another):</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row</strong>: Main.</td>
<td></td>
</tr>
<tr>
<td><strong>Trigger</strong>: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error.</td>
<td></td>
</tr>
</tbody>
</table>

| **Row**: Main; Reject; Iterate. |
| **Trigger**: Run if; On Subjob Ok; On Subjob Error; On component Ok; On Component Error; Synchronize; Parallelize. |

For further information regarding connections, see *Talend Studio User Guide*. |
Scenario: Restoring files from bin

This scenario describes a Job that iterates on a list of files in a directory, copies each file to a defined target directory, and then removes the copied files from the source directory.

![Diagram of tFileList and tFileCopy components with Iterate link]

Procedure

1. Create a new Job and add a tFileList component and a tFileCopy component by typing their names in the design workspace or dropping them from the Palette.
2. Connect tFileList to tFileCopy using a Row > Iterate link.
3. Double-click tFileList to open its Basic settings view.
   - In the Directory field, browse to or type in the directory to iterated upon.
4. Double-click tFileCopy to open its Basic settings view.
   - In the File Name field, use a global map expression to access the current file path from the tFileList component.
   - In the Destination directory field, set the target directory.
   - Enable options for removing source files and creating directories if they don’t exist.
6. In the **File Name** field, press **Ctrl+Space** to access the global variable list and select the `tFileList_1.CURRENT_FILEPATH` variable from the list to fill the field with `(String)globalMap.get("tFileList_1.CURRENT_FILEPATH")`.

7. In the **Destination directory** field, browse to or type in the directory to copy each file to.

8. Select the **Remove source file** check box to get rid of the files that have been copied.

9. Press **Ctrl+S** to save your Job and press **F6** to execute it.

   All the files in the defined source directory are copied to the destination directory and are removed from the source directory.
tFileDelete

Deletes files from a given directory.

**tFileDelete Standard properties**

These properties are used to configure tFileDelete running in the Standard Job framework.

The Standard tFileDelete component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>Path to the file to be deleted. This field is hidden when you select the Delete folder check box or the Delete file or folder check box.</td>
</tr>
<tr>
<td>Directory</td>
<td>Path to the folder to be deleted. This field is available only when you select the Delete folder check box.</td>
</tr>
<tr>
<td>File or directory to delete</td>
<td>Enter the path to the file or to the folder you want to delete. This field is available only when you select the Delete file or folder check box.</td>
</tr>
<tr>
<td>Fail on error</td>
<td>Select this check box to prevent the main Job from being executed if an error occurs, for example, if the file to be deleted does not exist.</td>
</tr>
<tr>
<td>Delete Folder</td>
<td>Select this check box to display the Directory field, where you can indicate the path the folder to be deleted.</td>
</tr>
<tr>
<td>Delete file or folder</td>
<td>Select this check box to display the File or directory to delete field, where you can indicate the path to the file or to the folder you want to delete.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE_PATH</td>
<td>the path to the deleted file or folder. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>CURRENT_STATUS</td>
<td>the execution result of the component. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions</td>
</tr>
</tbody>
</table>
Usage

Usage rule

This component can be used as standalone component.

Connections

Outgoing links (from this component to another):

- **Row**: Main.
- **Trigger**: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error.

Incoming links (from one component to this one):

- **Row**: Main; Reject; Iterate.
- **Trigger**: Run if; On Subjob Ok; On Subjob Error; On component Ok; On Component Error; Synchronize; Parallelize.

For further information regarding connections, see *Talend Studio User Guide*.

Scenario: Deleting files

This very simple scenario describes a Job deleting files from a given directory.

Procedure

1. Drop the following components: **tFileList, tFileDelete, tJava** from the **Palette** to the design workspace.
2. In the **tFileList Basic settings**, set the directory to loop on in the **Directory** field.
3. The filemask is "*.txt" and no case check is to carry out.

4. In the **tFileDelete** Basic settings panel, set the **File Name** field in order for the current file in selection in the **tFileList** component be deleted. This delete all files contained in the directory, as specified earlier.

5. press **Ctrl+Space bar** to access the list of global variables. In Java, the relevant variable to collect the current file is: `((String)globalMap.get("tFileList_1_CURRENT_FILEPATH")).`

6. Then in the **tJava** component, define the message to be displayed in the standard output (Run console). In this Java use case, type in the Code field, the following script: `System.out.println((String)globalMap.get("tFileList_1_CURRENT_FILE")+ " has been deleted!");`

7. Then save your Job and press **F6** to run it.

**Results**

The message set in the **tJava** component displays in the log, for each file that has been deleted through the **tFileDelete** component.
tFileExist

Checks if a file exists or not.

**tFileExist Standard properties**

These properties are used to configure tFileExist running in the Standard Job framework.

The Standard tFileExist component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>File Name</strong></th>
<th>Path to the file you want to check if it exists or not.</th>
</tr>
</thead>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>tStatCatcher Statistics</strong></th>
<th>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</th>
</tr>
</thead>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>EXISTS</strong>: the result of whether a specified file exists. This is a Flow variable and it returns a boolean.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>FILENAME</strong>: the name of the file processed. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

<table>
<thead>
<tr>
<th><strong>Usage rule</strong></th>
<th>This component can be used as standalone component.</th>
</tr>
</thead>
</table>

**Connections**

Outgoing links (from this component to another):

**Trigger**: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error.
Scenario: Checking for the presence of a file and creating it if it does not exist

This scenario describes a simple Job that: checks if a given file exists, displays a graphical message to confirm that the file does not exist, reads the input data in another given file and writes it in an output delimited file.

A dialog box appears to confirm that the file does not exists.

Click OK to close the dialog box and continue the Job execution process. The missing file, file1 in this scenario, got written in a delimited file in the defined place.

Dropping and linking the components

Procedure

1. Drop the following components from the Palette onto the design workspace: tFileExist, tFileInputDelimited, tFileOutputDelimited, and tMsgBox.
2. Connect tFileExist to tFileInputDelimited using an OnSubjobOk and to tMsgBox using a Run If link.
3. Connect tFileInputDelimited to tFileOutputDelimited using a Row Main link.

Configuring the components

Procedure

1. In the design workspace, select tFileExist and click the Component tab to define its basic settings.
2. In the **File name** field, enter the file path or browse to the file you want to check if it exists or not.

3. In the design workspace, select **tFileInputDelimited** and click the **Component** tab to define its basic settings.

4. Browse to the input file you want to read to fill out the **File Name** field.

   **Warning:**
   If the path of the file contains some accented characters, you will get an error message when executing your Job.

5. Set the row and field separators in their corresponding fields.

6. Set the header, footer and number of processed rows as needed. In this scenario, there is one header in our table.

7. Set **Schema** to **Built-in** and click the **Edit schema** button to define the data to pass on to the **tFileOutputDelimited** component. Define the data present in the file to read, file2 in this scenario.
   For more information about schema types, see **Talend Studio User Guide**.
   
   The schema in file2 consists of five columns: **Num, Ref, Price, Quant, and tax**.

8. In the design workspace, select the **tFileOutputDelimited** component.

9. Click the **Component** tab to define the basic settings of **tFileOutputDelimited**.
10. Set property type to **Built-in**.

11. In the **File name** field, press **Ctrl+Space** to access the variable list and select the global variable FILENAME.

12. Set the row and field separators in their corresponding fields.

13. Select the **Include Header** check box as file2 in this scenario includes a header.

14. Set **Schema** to **Built-in** and click **Sync columns** to synchronize the output file schema (file1) with the input file schema (file2).

15. In the design workspace, select the **tMsgBox** component.

16. Click the **Component** tab to define the basic settings of **tMsgBox**.

17. Click the **If** link to display its properties in the **Basic settings** view.

18. In the **Condition** panel, press **Ctrl+Space** to access the variable list and select the global variable EXISTS. Type an exclamation mark before the variable to negate the meaning of the variable.
Saving and executing the Job

Procedure

1. Press **Ctrl+S** to save your Job.
2. Press **F6** or click the **Run** button in the **Run** tab to execute it.
tFileFetch

Retrieves a file through the given protocol (HTTP, HTTPS, FTP, or SMB).

**tFileFetch Standard properties**

These properties are used to configure tFileFetch running in the Standard Job framework.
The Standard tFileFetch component belongs to the Internet family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Protocol</strong></td>
<td>Select the protocol you want to use from the list and fill in the corresponding fields: http, https, ftp, smb. The properties differ slightly depending on the type of protocol selected. The additional fields are defined in this table, after the basic settings.</td>
</tr>
<tr>
<td><strong>URI</strong></td>
<td>Type in the URI of the site from which the file is to be fetched.</td>
</tr>
<tr>
<td><strong>Use cache to save resource</strong></td>
<td>Select this check box to save the data in the cache. This option allows you to process the file data flow (in streaming mode) without saving it on your drive. This is faster and improves performance.</td>
</tr>
<tr>
<td><strong>Domain</strong></td>
<td>Enter the Microsoft server domain name. Available for the smb protocol.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Enter the authentication information required to access the server. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. Available for the smb protocol.</td>
</tr>
<tr>
<td><strong>Destination Directory</strong></td>
<td>Browse to the destination folder where the file fetched is to be placed.</td>
</tr>
<tr>
<td><strong>Destination Filename</strong></td>
<td>Enter a new name for the file fetched.</td>
</tr>
<tr>
<td><strong>Create full path according to URI</strong></td>
<td>It allows you to reproduce the URI directory path. To save the file at the root of your destination directory, clear the check box. Available for the http, https and ftp protocols.</td>
</tr>
<tr>
<td><strong>Add header</strong></td>
<td>Select this check box if you want to add one or more HTTP request headers as fetch conditions. In the Headers table, enter the name(s) of the HTTP header</td>
</tr>
</tbody>
</table>
parameter(s) in the **Name** field and the corresponding value(s) in the **Value** field.
Available for the **http** and **https** protocols.

### POST method
This check box is selected by default. It allows you to use the POST method. In the **Parameters** table, enter the name of the variable(s) in the **Name** field and the corresponding value in the **Value** field.
Clear the check box if you want to use the GET method.
Available for the **http** and **https** protocols.

### Die on error
Clear this check box to skip the rows in error and to complete the process for the error free rows
Available for the **http**, **https** and **ftp** protocols.

### Read Cookie
Select this check box for **tFileFetch** to load a web authentication cookie.
Available for the **http**, **https**, **ftp** and **smb** protocols.

### Save Cookie
Select this check box to save the web page authentication cookie. This means you will not have to log on to the same web site in the future.
Available for the **http**, **https**, **ftp** and **smb** protocols.

### Cookie file
Type in the full path to the file which you want to use to save the cookie or click […] and browse to the desired file to save the cookie.
Available for the **http**, **https**, **ftp** and **smb** protocols.

### Cookie policy
Choose a cookie policy from this drop-down list. Four options are available, **BROWSER_COMPATIBILITY**, **DEFAULT**, **NETSCAPE** and **RFC_2109**.
Available for the **http**, **https**, **ftp** and **smb** protocols.

### Single cookie header
Check this box to put all cookies into one request header for maximum compatibility among different servers.
Available for the **http**, **https**, **ftp** and **smb** protocols.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>tStatCatcher Statistics</strong></th>
<th>Select this check box to collect the log data at each component level.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timeout</strong></td>
<td>Enter the number of milliseconds after which the protocol connection should close.</td>
</tr>
<tr>
<td></td>
<td>Available for the <strong>http</strong> and <strong>https</strong> protocols.</td>
</tr>
<tr>
<td><strong>Print response to console</strong></td>
<td>Select this check box to print the server response in the console.</td>
</tr>
<tr>
<td></td>
<td>Available for the <strong>http</strong> and <strong>https</strong> protocols.</td>
</tr>
</tbody>
</table>
### Upload file

Select this check box to upload one or more files to the server. Then in the Files table displayed, click the [+] button to add the file(s) to upload and define the following parameters for each file:

- **Name**: the new name of the file after being uploaded, between double quotation marks.
- **File**: the full path of the file to upload, e.g. `D:/filefetch.txt`.
- **Content-Type**: the content type of the file to upload. The default value is "application/octet-stream".
- **Charset**: the character set of the file to upload. The default value is "ISO-8859-1".

Available for the **http** and **https** protocols.

### Enable proxy server

Select this check box if you are connecting via a proxy and complete the fields which follow with the relevant information.

Available for the **http**, **https** and **ftp** protocols.

### Enable NTLM Credentials

Select this check box if you are using an NTLM authentication protocol.

- **Domain**: The client domain name.
- **Host**: The client’s IP address.

Available for the **http** and **https** protocols.

### Need authentication

Select this check box and enter the username and password in the relevant fields, if they are required to access the protocol.

Available for the **http** and **https** protocols.

### Support redirection

Select this check box to repeat the redirection request until redirection is successful and the file can be retrieved.

Available for the **http**, **https** and **ftp** protocols.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td><strong>INPUT_STREAM</strong></td>
<td>the content of the file being fetched. This is a Flow variable and it returns an InputStream. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
</tbody>
</table>

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | This component is generally used as a start component to feed the input flow of a Job and is often connected to the Job using an **OnSubjobOk** or **OnComponentOk** link, depending on the context. |
| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |

### Scenario 1: Fetching data through HTTP

This scenario describes a three-component Job which retrieves a file from an HTTP website, reads data from the fetched file and displays the data on the console.

![Diagram](image)

**Dropping and linking components**

**Procedure**

1. Drop a **tFileFetch**, a **tFileInputDelimited** and a **tLogRow** onto your design workspace.
2. Link **tFileFetch** to **tFileInputDelimited** using a **Trigger > On Subjob Ok** or **On Component Ok** connection.
3. Link **tFileInputDelimited** to **tLogRow** using a **Row > Main** connection.
Configuring the components

Procedure

1. Double-click **tFileFetch** to open its **Basic settings** view.

   ![tFileFetch](image)

   - **Protocol**: http
   - **URI**: "http://talendforge.org/file_fetch.txt"
   - **Use cache to save the resource**: unchecked
   - **Destination directory**: "D:/Output"
   - **Create full path according to URI**: unchecked
   - **Destination filename**: "new.txt"
   - **Add header**: unchecked
   - **POST method**: unchecked
   - **Die on error**: unchecked
   - **Read cookie**: unchecked
   - **Save cookie**: unchecked

2. Select the protocol you want to use from the list. Here, **http** is selected.

3. In the **URI** field, type in the URI where the file to be fetched can be retrieved from. You can paste the URI directly in your browser to view the data in the file.

4. In the **Destination directory** field, browse to the folder where the fetched file is to be stored. In this example, it is "D:/Output".

5. In the **Destination filename** field, type in a new name for the file if you want it to be changed. In this example, "new.txt".

6. If needed, select the **Add header** check box and define one or more HTTP request headers as fetch conditions. For example, to fetch the file only if it has been modified since 19:43:31 GMT, October 29, 1994, fill in the **Name** and **Value** fields with "If-Modified-Since" and "Sat, 29 Oct 1994 19:43:31 GMT" respectively in the **Headers** table. For details about HTTP request header definitions, see Header Field Definitions.

7. Double-click **tFileInputDelimited** to open its **Basic settings** view.

   ![tFileInputDelimited](image)

   - **Property Type**: Built-In
   - **File name/Stream**: "D:/Output/new.txt"
   - **Row Separator**: "\n"
   - **Field Separator**: "," (with a custom setting)
   - **CSV options**: unchecked
   - **Header**: 0
   - **Footer**: 0
   - **Limit**: unchecked
   - **Schema**: Built-In
   - **Skip empty rows**: checked
   - **Uncompress as zip file**: unchecked
   - **Die on error**: unchecked

8. In the **File name** field, type in the full path to the fetched file which had been stored locally.
9. Click the [...] button next to **Edit schema** to open the **[Schema]** dialog box. In this example, add one column **output** to store the data from the fetched file.

10. Leave other settings as they are.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Press **F6** or click **Run** on the **Run** tab to execute the Job.

```
Starting job test at 17:14 10/01/2014.
[statistics] connecting to socket on port 3656
[statistics] connected
Well done
[statistics] disconnected
Job test ended at 17:14 10/01/2014. [exit code=0]
```

The data of the fetched file is displayed on the console.

**Scenario 2: Reusing stored cookie to fetch files through HTTP**

This scenario describes a two-component Job which logs in a given HTTP website and then using cookie stored in a user-defined local directory, fetches data from this website.
Dropping and linking components

Procedure
1. Drop two `tFileFetch` components onto your design workspace.
2. Link the two components as subjobs using a `Trigger > On Subjob Ok` connection.

Configuring the components

Configuring the first subjob

Procedure
1. Double click `tFileFetch_1` to open its component view.
2. Select the protocol you want to use from the **Protocol** list. Here, we use the **https** protocol.

3. In the **URI** field, type in the URI through which you can log in the website and fetch the web page accordingly. In this example, the URI is **https://www.codeproject.com/script/Membership/LogOn.aspx?download=true**.

4. In the **Destination directory** field, browse to the folder where the fetched web page is to be stored. This folder will be created on the fly if it does not exist. In this example, type in **D:/download**.

5. In the **Destination Filename** field, type in a new name for the file if you want it to be changed. In this example, **codeproject.html**.

6. Under the **Parameters** table, click the plus button to add two rows and fill in the credentials for accessing the desired website.

   In the **Name** column, type in a new name respectively for the two rows. In this example, they are **Email** and **Password**, which are required by the website you are logging in.

   In the **Value** column, type in the authentication information.

7. Select the **Save cookie** check box.

8. In the **Cookie file** field, type in the full path to the file which you want to use to save the cookie. In this example, it is **D:/download/cookie**.

9. Click **Advanced settings** to open its view.

10. Select the **Support redirection** check box so that the redirection request will be repeated until the redirection is successful.

    **Configuring the second subjob**

   **Procedure**

1. Double-click **tFileFetch_2** to open its **Component** view.
2. From the **Protocol** list, select **http**.

3. In the **URI** field, type in the address from which you fetch the files of your interest. In this example, the address is http://www.codeproject.com/script/articles/download.aspx?file=/KB/DLL/File_List_Downloader/FLD02June2011_Source.zip&rp=http://www.codeproject.com/Articles/203991/File-List-Downloader.

4. In the **Destination directory** field, type in the directory or browse to the folder where you want to store the fetched files. This folder can be automatically created if it does not exist yet during the execution process. In this example, type in `D:/download`.

5. In the **Destination Filename** field, type in a new name for the file if you want it to be changed. In this example, `source.zip`.

6. Clear the **POST method** check box to deactivate the **Parameters** table.

7. Select the **Read cookie** check box.

8. In the **Cookie file** field, browse to the file which is used to save the cookie. In this example, it is `D:/download/cookie`.

### Saving and executing the Job

**Procedure**

1. Press **Ctrl+S** to save your Job.

2. Press **F6** or click **Run** on the **Run** tab to execute the Job.

   Then, go to the local directory `D:/download` to check the downloaded file.

   ![Directory Contents](image)
Related scenario

For an example of transferring data in streaming mode, see Scenario 2: Reading data from a remote file in streaming mode on page 1015
tFileInputARFF

Reads an ARFF file row by row to split them up into fields and then sends the fields as defined in the schema to the next component.

**tFileInputARFF Standard properties**

These properties are used to configure tFileInputARFF running in the Standard Job framework.

The Standard tFileInputARFF component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file where the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

**File Name**

Name and path of the ARFF file and/or variable to be processed.

For further information about how to define and use a variable in a Job, see *Talend Studio User Guide*.

**Schema** and **Edit Schema**

A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.
### Built-in
The schema will be created and stored locally for this component only. Related topic: see [Talend Studio User Guide](https://help.talend.com).

### Repository
The schema already exists and is stored in the Repository, hence can be reused in various projects and Job flowcharts. Related topic: see [Talend Studio User Guide](https://help.talend.com).

### Advanced settings

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Select the encoding type from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for DB data handling.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <a href="https://help.talend.com">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Use this component to read a file and separate the fields with the specified separator.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the <strong>Install</strong> button on the <strong>Component</strong> tab view. You can also find out and add all missing JARs easily on the <strong>Modules</strong> tab in the <strong>Integration</strong> perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>
Scenario: Display the content of a ARFF file

This scenario describes a two-component Job in which the rows of an ARFF file are read, the delimited data is selected and the output is displayed in the Run view.

An ARFF file looks like the following:

```
@relation vote
@attribute 'handicapped-infants' { 'n', 'y'}
@attribute 'water-project-cost-sharing' { 'n', 'y'}
@attribute 'adoption-of-the-budget-resolution' { 'n', 'y'}
@attribute 'physician-fee-freeze' { 'n', 'y'}
@attribute 'el-salvador-aid' { 'n', 'y'}
@attribute 'religious-groups-in-schools' { 'n', 'y'}
@attribute 'anti-satellite-test-ban' { 'n', 'y'}
@attribute 'aid-to-nicaraguan-contras' { 'n', 'y'}
@attribute 'mx-missile' { 'n', 'y'}
@attribute 'immigration' { 'n', 'y'}
@attribute 'synfuels-corporation-cutback' { 'n', 'y'}
@attribute 'education-spending' { 'n', 'y'}
@attribute 'superc fund-right-to-sue' { 'n', 'y'}
@attribute 'crime' { 'n', 'y'}
@attribute 'duty-free-exports' { 'n', 'y'}
@attribute 'export-administration-act-south-africa' { 'n', 'y'}
@attribute 'Class' { 'democrat', 'republican'}
@data
'n','y','n','y','y','y','n','n','n','y','?','y','y','y','n','n','y','y','n','y','?','republican'
'n','y','n','y','y','y','y','y','n','n','n','n','n','n','n','y','y','y','y','n','?','republican'
?, 'y','y','?','y','y','n','n','n','n','n','y','n','y','n','n','n','n','y','y','n','n','n','democrat'
```

It is generally made of two parts. The first part describes the data structure, that is to say the rows which begin by `@attribute` and the second part comprises the raw data, which follows the expression `@data`.

Dropping and linking components

**Procedure**

1. Drop the tFileInputARFF component from the Palette onto the workspace.
2. In the same way, drop the tLogRow component.
3. Right-click the tFileInputARFF and select Row > Main in the menu. Then, drag the link to the tLogRow, and click it. The link is created and appears.

Configuring the components

**Procedure**

1. Double-click the tFileInputARFF.
2. In the **Component** view, in the **File Name** field, browse your directory in order to select your .arff file.

3. In the **Schema** field, select **Built-In**.

4. Click the [...] button next to **Edit schema** to add column descriptions corresponding to the file to be read.

5. Click on the [ ] button as many times as required to create the number of columns required, according to the source file. Name the columns as follows.

6. For every column, the **Nullable** check box is selected by default. Leave the check boxes selected, for all of the columns.

7. Click **OK**.

8. In the workspace, double-click the **tLogRow** to display its **Component** view.
9. Click the [...] button next to Edit schema to check that the schema has been propagated. If not, click the Sync columns button.

**Saving and executing the Job**

**Procedure**

1. Press Ctrl+S to save your Job.
2. Press F6 to execute your Job.

The console displays the data contained in the ARFF file, delimited using a vertical line (the default separator).
tFileInputDelimited

Reads a delimited file row by row to split them up into fields and then sends the fields as defined in the schema to the next component.

**tFileInputDelimited Standard properties**

These properties are used to configure tFileInputDelimited running in the Standard Job framework. The Standard tFileInputDelimited component belongs to the File family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-In or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored.</td>
</tr>
<tr>
<td>File Name/Stream</td>
<td><strong>File name</strong>: Name and path of the file to be processed. <strong>Stream</strong>: The data flow to be processed. The data must be added to the flow in order for tFileInputDelimited to fetch these data via the corresponding representative variable. This variable could be already pre-defined in your Studio or provided by the context or the components you are using along with this component; otherwise, you could define it manually and use it according to the design of your Job, for example, using tJava or tJavaFlex. In order to avoid the inconvenience of hand writing, you could select the variable of interest from the auto-completion list (Ctrl+Space) to fill the current field on condition that this variable has been properly defined. Related topic to the available variables: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Row separator</td>
<td>The separator used to identify the end of a row.</td>
</tr>
<tr>
<td>Field separator</td>
<td>Enter character, string or regular expression to separate fields for the transferred data.</td>
</tr>
<tr>
<td>CSV options</td>
<td>Select this check box to specify the following CSV parameters: <strong>Escape char</strong>: enter the escape character between double quotation marks. For example, you need to type in &quot;&quot; if the escape character () is used as the escape character. <strong>Text enclosure</strong>: enter the enclosure character (only one character) between double quotation</td>
</tr>
</tbody>
</table>
marks. For example, "" " needs to be entered when double quotation marks (‘) are used as the enclosure character.

<table>
<thead>
<tr>
<th>Header</th>
<th>Enter the number of rows to be skipped in the beginning of file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Footer</td>
<td>Number of rows to be skipped at the end of the file.</td>
</tr>
<tr>
<td>Limit</td>
<td>Maximum number of rows to be processed. If Limit = 0, no row is read or processed.</td>
</tr>
</tbody>
</table>

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.

Note that if the input value of any non-nullable primitive field is null, the row of data including that field will be rejected.

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

**Skip empty rows**

Select this check box to skip the empty rows.

**Uncompress as zip file**

Select this check box to uncompress the input file.

**Die on error**

Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a **Row > Reject** link.

To catch the `FileNotFoundException`, you also need to select this check box.
Advanced settings

<table>
<thead>
<tr>
<th>Advanced separator (for numbers)</th>
<th>Select this check box to change the separator used for numbers. By default, the thousands separator is a comma (,) and the decimal separator is a period (.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract lines at random</td>
<td>Select this check box to set the number of lines to be extracted randomly.</td>
</tr>
<tr>
<td>Encoding</td>
<td>Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for database data handling.</td>
</tr>
<tr>
<td>Trim all column</td>
<td>Select this check box to remove the leading and trailing whitespaces from all columns. When this check box is cleared, the <strong>Check column to trim</strong> table is displayed, which lets you select particular columns to trim.</td>
</tr>
<tr>
<td>Check each row structure against schema</td>
<td>Select this check box to check whether the total number of columns in each row is consistent with the schema. If not consistent, an error message will be displayed on the console.</td>
</tr>
<tr>
<td>Check date</td>
<td>Select this check box to check the date format strictly against the input schema.</td>
</tr>
<tr>
<td>Check columns to trim</td>
<td>This table is filled automatically with the schema being used. Select the check box(es) corresponding to the column(s) to be trimmed.</td>
</tr>
<tr>
<td>Split row before field</td>
<td>Select this check box to split rows before splitting fields.</td>
</tr>
</tbody>
</table>

**Permit hexadecimal (0xNNN) or octal (0NNNN) for numeric types**
Select this check box if any of your numeric types (long, integer, short, or byte type), will be parsed from a hexadecimal or octal string.
In the table that displays, select the check box next to the column or columns of interest to transform the input string of each selected column to the type defined in the schema.
Select the **Permit hexadecimal or octal** check box to select all the columns.
This table appears only when the **Permit hexadecimal (0xNNN) or octal (0NNNN) for numeric types** check box is selected.

**tStatCatcher Statistics**
Select this check box to gather the processing metadata at the Job level as well as at each component level.

Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions</td>
</tr>
</tbody>
</table>
only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

Use this component to read a file and separate fields contained in this file using a defined separator. It allows you to create a data flow using a Row > Main link or via a Row > Reject link in which case the data is filtered by data that does not correspond to the type defined. For further information, please see Procedure on page 971.

Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Scenario: Delimited file content display

The following scenario creates a two-component Job, which aims at reading each row of a file, selecting delimited data and displaying the output in the Run log console.

Dropping and linking components

Procedure

1. Drop a tFileInputDelimited component and a tLogRow component from the Palette to the design workspace.
2. Right-click on the tFileInputDelimited component and select Row > Main. Then drag it onto the tLogRow component and release when the plug symbol shows up.
**Configuring the components**

**Procedure**

1. Select the `tFileInputDelimited` component again, and define its Basic settings:

   ![Picture of tFileInputDelimited component settings]

   - **Property Type**: Repository
   - **File Name**: `C:/Input/owners.csv`
   - **Row Separator**: `\n`
   - **Field Separator**: `*`
   - **Header**: 1
   - **Footer**: 0
   - **Limit**: 50
   - **Schema**: Repository (Built-in)
   - **Encoding**: `tLogRow` on page 1988
   - **Print schema column name in front of each value**: selected

2. Fill in a path to the file in the **File Name** field. This field is mandatory.

   - **Warning**: If the path of the file contains some accented characters, you will get an error message when executing your Job.

3. Define the **Row separator** allowing to identify the end of a row. Then define the **Field separator** used to delimit fields in a row.

4. In this scenario, the header and footer limits are not set. And the **Limit** number of processed rows is set on 50.

5. Set the **Schema** as either a local (Built-in) or a remotely managed (Repository) to define the data to pass on to the `tLogRow` component.

6. You can load and/or edit the schema via the **Edit Schema** function.

   Related topics: see *Talend Studio User Guide*.

7. Enter the encoding standard the input file is encoded in. This setting is meant to ensure encoding consistency throughout all input and output files.

8. Select the `tLogRow` and define the **Field separator** to use for the output display. Related topic: `tLogRow` on page 1988.

9. Select the **Print schema column name in front of each value** check box to retrieve the column labels in the output displayed.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.

2. Go to **Run** tab, and click on **Run** to execute the Job.

   The file is read row by row and the extracted fields are displayed on the **Run** log as defined in both components **Basic settings**.
Scenario 2: Reading data from a remote file in streaming mode

This scenario describes a four component Job used to fetch data from a voluminous file almost as soon as it has been read. The data is displayed in the Run view. The advantage of this technique is that you do not have to wait for the entire file to be downloaded, before viewing the data.

Dropping and linking components

Procedure

1. Drop the following components onto the workspace: tFileFetch, tSleep, tFileInputDelimited, and tLogRow.

2. Connect tSleep and tFileInputDelimited using a Trigger > OnComponentOk link and connect tFileInputDelimited to tLogRow using a Row > Main link.

Configuring the components

Procedure

1. Double-click tFileFetch to display the Basic settings tab in the Component view and set the properties.
2. From the Protocol list, select the appropriate protocol to access the server on which your data is stored.

3. In the URI field, enter the URI required to access the server on which your file is stored.

4. Select the Use cache to save the resource check box to add your file data to the cache memory. This option allows you to use the streaming mode to transfer the data.

5. In the workspace, click tSleep to display the Basic settings tab in the Component view and set the properties.

   By default, tSleep’s Pause field is set to 1 second. Do not change this setting. It pauses the second Job in order to give the first Job, containing tFileFetch, the time to read the file data.

6. In the workspace, double-click tFileInputDelimited to display its Basic settings tab in the Component view and set the properties.

7. In the File name/Stream field:
   - Delete the default content.
   - Press Ctrl+Space to view the variables available for this component.
   - Select tFileFetch_1_INPUT_STREAM from the auto-completion list, to add the following variable to the Filename field: `((java.io.InputStream)globalMap.get("tFileFetch_1_INPUT_STREAM"))`. 
8. From the **Schema** list, select **Built-in** and click [...] next to the **Edit schema** field to describe the structure of the file that you want to fetch. The *US_Employees* file is composed of six columns: *ID*, *Employee*, *Age*, *Address*, *State*, *EntryDate*.

   Click [+] to add the six columns and set them as indicated in the above screenshot. Click **OK**.

![Schema of tFileInputDelimited_1](image)

9. In the workspace, double-click **tLogRow** to display its **Basic settings** in the **Component** view and click **Sync Columns** to ensure that the schema structure is properly retrieved from the preceding component.

**Configuring Job execution and executing the Job**

**Procedure**

1. Click the **Job** tab and then on the **Extra** view.

![tFileInputDelimited 0.1](image)

2. Select the **Multi thread execution** check box in order to run the two Jobs at the same time. Bear in mind that the second Job has a one second delay according to the properties set in **tSleep**. This option allows you to fetch the data almost as soon as it is read by **tFileFetch**, thanks to the **tFileDelimited** component.

3. Save the Job and press **F6** to run it.
The data is displayed in the console as almost as soon as it is read.
tFileInputExcel

Reads an Excel file row by row to split them up into fields using regular expressions and then sends the fields as defined in the schema to the next component.

**tFileInputExcel Standard properties**

These properties are used to configure tFileInputExcel running in the Standard Job framework.

The Standard tFileInputExcel component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file where the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

| Read excel2007 file format (.xlsx) | Select this check box to read the .xlsx file of Excel 2007. |

| File Name/Stream | **File name**: Name of the file and/or the variable to be processed. |
|------------------| **Stream**: Data flow to be processed. The data must be added to the flow in order to be collected by tFileInputExcel via the INPUT_STREAM variable in the auto-completion list (Ctrl+Space). |
|                  | For further information about how to define and use a variable in a Job, see Talend Studio User Guide. |

| All sheets | Select this check box to process all sheets of the Excel file. |

| Sheet list | Click the plus button to add as many lines as needed to the list of the excel sheets to be processed: |
|------------| **Sheet (name or position)**: enter the name or position of the excel sheet to be processed. |
|            | **Use Regex**: select this check box if you want to use a regular expression to filter the sheets to process. |

| Header | Enter the number of rows to be skipped in the beginning of file. |
### Footer

Number of records to be skipped at the end of the file.

### Limit

Maximum number of lines to be processed.

### Affect each sheet(header&footer)

Select this check box if you want to apply the parameters set in the **Header** and **Footer** fields to all excel sheets to be processed.

### Die on error

Select this check box to stop the execution of the Job when an error occurs. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can collect the rows on error using a **Row > Reject** link.

### First column and Last column

Define the range of the columns to be processed through setting the first and last columns in the **First column** and **Last column** fields respectively.

### Schema and Edit Schema

A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

- **Built-in**: The schema will be created and stored locally for this component only. Related topic: see **Talend Studio User Guide**.

- **Repository**: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job flowcharts. Related topic: see **Talend Studio User Guide**.

### Advanced settings

**Advanced separator**

Select this check box to change the used data separators.

**Trim all columns**

Select this check box to remove the leading and trailing whitespaces from all columns. When this check box is cleared, the **Check column to trim** table is displayed, which lets you select particular columns to trim.
<table>
<thead>
<tr>
<th><strong>Check column to trim</strong></th>
<th>This table is filled automatically with the schema being used. Select the check box(es) corresponding to the column(s) to be trimmed.</th>
</tr>
</thead>
</table>
| **Convert date column to string** | Available when Read excel2007 file format (xlsx) is selected in the Basic settings view.  
Select this check box to show the table Check need convert date column. Here you can parse the string columns that contain date values based on the given date pattern.  
**Column:** all the columns available in the schema of the source .xlsx file.  
**Convert:** select this check box to choose all the columns for conversion (only if they are all of the string type). You can also select the individual check box next to each column for conversion.  
**Date pattern:** set the date format here. |
| **Encoding** | Select the encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling. |
| **Read real values for numbers** | Select this check box to read numbers in real values. This check box becomes unavailable when you select Read excel2007 file format (xlsx) in the Basic settings view. |
| **Stop reading on encountering empty rows** | Select this check box to ignore the empty line encountered and, if there are any, the lines that follow this empty line. This check box becomes unavailable when you select Read excel2007 file format (xlsx) in the Basic settings view. |
| **Generation mode** | Available when Read excel2007 file format (xlsx) is selected in the Basic settings view. Select the mode used to read the Excel 2007 file.  
- **Less memory consumed for large excel (Event mode):** used for large file. This is a memory-saving mode to read the Excel 2007 file as a flow. With this mode selected, the data will be extracted with the format symbol, for example, the percent symbol % and the currency symbol $.  
- **Memory-consuming (User mode):** used for small file. It needs much memory. With this mode selected, the pure data without the format symbol will be extracted. |
| **Don't validate the cells** | Select this check box to in order not to validate data. This check box becomes unavailable when you select Read excel2007 file format (xlsx) in the Basic settings view. |
| **Ignore the warning** | Select this check box to ignore all warnings generated to indicate errors in the Excel file. This check box |
becomes unavailable when you select **Read excel2007 file format (xlsx)** in the **Basic settings** view.

### tStatCatcher Statistics

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</th>
</tr>
</thead>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>CURRENT_SHEET</strong>: the name of the sheet being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Use this component to read an Excel file and to output the data separately depending on the schemas identified in the file. You can use a <strong>Row &gt; Reject</strong> link to filter the data which doesn't correspond to the type defined. For an example of how to use these two links, see <strong>Procedure</strong> on page 971.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the <strong>Install</strong> button on the <strong>Component</strong> tab view. You can also find out and add all missing JARs easily on the <strong>Modules</strong> tab in the <strong>Integration</strong> perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

### Related scenarios

No scenario is available for the Standard version of this component yet.
tFileInputFullRow

Reads a file row by row and sends complete rows of data as defined in the schema to the next component via a Row link.

**tFileInputFullRow Standard properties**

These properties are used to configure tFileInputFullRow running in the Standard Job framework.

The Standard tFileInputFullRow component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. |
| **Built-In**    | You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*. |
| **Repository**  | You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*. |
| **File Name**   | Specify the path to the file to be processed. |
| **Row separator** | The separator used to identify the end of a row. |
| **Header**      | Enter the number of rows to be skipped in the beginning of file. |
| **Footer**      | Enter the number of rows to be skipped at the end of the file. |
**Limit** | Enter the maximum number of rows to be processed. If the value is set to 0, no row is read or processed.

**Skip empty rows** | Select this check box to skip the empty rows.

### Advanced settings

**Encoding** | Select the encoding from the list or select Custom and define it manually. This field is compulsory for database data handling.

**Extract lines at random** | Select this check box to set the number of lines to be extracted randomly.

**tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

### Global Variables

**Global Variables**

- **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

**Usage rule** | Use this component to read full rows in delimited files that can get very large.

### Scenario: Reading full rows in a delimited file

The following scenario creates a two-component Job that aims at reading complete rows in the delimited file *states.csv* and displaying the rows on the console.

The content of the file *states.csv* that holds ten rows of data is as follows:

<table>
<thead>
<tr>
<th>StateID</th>
<th>StateName</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alabama</td>
</tr>
<tr>
<td>2</td>
<td>Alaska</td>
</tr>
<tr>
<td>3</td>
<td>Arizona</td>
</tr>
</tbody>
</table>
Reading full rows in a delimited file

Procedure

1. Create a new Job and add a `tFileInputFullRow` component and a `tLogRow` component by typing their names in the design workspace or dropping them from the Palette.

2. Link the `tFileInputFullRow` component to the `tLogRow` component using a `Row > Main` connection.

3. Double-click the `tFileInputFullRow` component to open its `Basic settings` view on the Component tab.

4. Click the [...] button next to `Edit schema` to view the data to be passed onto the `tLogRow` component. Note that the schema is read-only and it consists of only one column `line`.

5. In the `File Name` field, browse to or enter the path to the file to be processed. In this scenario, it is `E:/states.csv`.

6. In the `Row Separator` field, enter the separator used to identify the end of a row. In this example, it is the default value `\n`.
7. In the **Header** field, enter 1 to skip the header row at the beginning of the file.

8. Double-click the **tLogRow** component to open its **Basic settings** view on the **Component** tab.

![tLogRow_1](image)

<table>
<thead>
<tr>
<th>Basic settings</th>
<th>Schema</th>
<th>Built-In</th>
<th>Edit schema</th>
<th>Sync columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced settings</td>
<td>Mode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic settings</td>
<td></td>
<td>Basic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>View</td>
<td>Table (print values in cells of a table)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vertical (each row is a key/value list)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Print content with log4j</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the **Mode** area, select **Table (print values in cells of a table)** for better readability of the result.

9. Press **Ctrl+S** to save your Job and then **F6** to execute it.

```plaintext
[statistics] connecting to socket on port 3617
[statistics] connected

<table>
<thead>
<tr>
<th>line</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Alabama</td>
</tr>
<tr>
<td>2: Alaska</td>
</tr>
<tr>
<td>3: Arizona</td>
</tr>
<tr>
<td>4: Arkansas</td>
</tr>
<tr>
<td>5: California</td>
</tr>
<tr>
<td>6: Colorado</td>
</tr>
<tr>
<td>7: Connecticut</td>
</tr>
<tr>
<td>8: Delaware</td>
</tr>
<tr>
<td>9: Florida</td>
</tr>
<tr>
<td>10: Georgia</td>
</tr>
</tbody>
</table>

[statistics] disconnected
```

As shown above, ten rows of data in the delimited file *states.csv* are read one by one, ignoring field separators, and the complete rows of data are displayed on the console.

To extract fields from rows, you must use **tExtractDelimitedFields**, **tExtractPositionalFields**, or **tExtractRegexFields**. For more information, see **tExtractDelimitedFields** on page 939, **tExtractPositionalFields** on page 959 and **tExtractRegexFields** on page 962.
tFileInputJSON

Extracts JSON data from a file and transfers the data to a file, a database table, etc.

**tFileInputJSON Standard properties**

These properties are used to configure tFileInputJSON running in the Standard Job framework.

The Standard tFileInputJSON component belongs to the Internet and the File families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either <strong>Built-In</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file where the properties are stored.</td>
</tr>
</tbody>
</table>

**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.

- **View schema**: choose this option to view the schema only.

- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Read By**

Select a way of extracting JSON data in the file.

- **JsonPath**: Extracts JSON data based on the JSONPath query. With this option selected, you need to select a JSONPath API version from the **API version** drop-
down list. It is recommended to read data by JSONPath in order to gain better performance.

- **Xpath**: Extracts JSON data based on the XPath query.
- **JsonPath without loop**: Extracts JSON data based on the JSONPath query without setting a loop node.

<table>
<thead>
<tr>
<th>Use Url</th>
<th>Select this check box to retrieve data directly from the Web.</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>Enter the URL path from which you will retrieve data. This field is available only when the Use Url check box is selected.</td>
</tr>
<tr>
<td>Filename</td>
<td>Specify the file from which you will retrieve data. This field is not visible if the Use Url check box is selected.</td>
</tr>
<tr>
<td>Loop Jsonpath query</td>
<td>Enter the path pointing to the node within the JSON field, on which the loop is based. Note if you have selected Xpath from the Read by drop-down list, the Loop Xpath query field is displayed instead.</td>
</tr>
<tr>
<td>Mapping</td>
<td>Complete this table to map the columns defined in the schema to the corresponding JSON nodes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Column</strong>: The <strong>Column</strong> cells are automatically filled with the defined schema column names.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Json query/JSONPath query</strong>: Specify the JSONPath node that holds the desired data. For more information about JSONPath expressions, see <a href="http://goessner.net/articles/JsonPath/">http://goessner.net/articles/JsonPath/</a>. This column is available only when <strong>JsonPath</strong> is selected from the <strong>Read By</strong> list.</td>
</tr>
<tr>
<td></td>
<td>• <strong>XPath query</strong>: Specify the XPath node that holds the desired data. This column is available only when <strong>Xpath</strong> is selected from the <strong>Read By</strong> list.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Get Nodes</strong>: Select this check box to extract the JSON data of all the nodes or select the check box next to a specific node to extract the data of that node. This column is available only when <strong>Xpath</strong> is selected from the <strong>Read By</strong> list.</td>
</tr>
<tr>
<td>Die on error</td>
<td>Select this check box to stop the execution of the Job when an error occurs. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can collect the rows on error using a <strong>Row &gt; Reject</strong> link.</td>
</tr>
</tbody>
</table>
### Advanced settings

<table>
<thead>
<tr>
<th>Advanced settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced separator (for numbers)</td>
<td>Select this check box to change the separator used for numbers. By default, the thousands separator is a comma (,) and the decimal separator is a period (.).</td>
</tr>
<tr>
<td><strong>Thousands separator</strong>: define separators for thousands.</td>
<td><strong>Decimal separator</strong>: define separators for decimals.</td>
</tr>
<tr>
<td>Validate date</td>
<td>Select this check box to check the date format strictly against the input schema. This check box is available only if the Read By XPath check box is selected.</td>
</tr>
<tr>
<td>Encoding</td>
<td>Select the encoding type from the list or select Custom and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component is a start component of a Job and always needs an output link.</td>
<td></td>
</tr>
</tbody>
</table>

### Scenario 1: Extracting JSON data from a file using JSONPath without setting a loop node

This scenario describes a two-component Job that extracts data from the JSON file Store.json by specifying the complete JSON path for each node of interest and displays the flat data extracted on the console.
The JSON file Store.json contains information about a department store and the content of the file is as follows:

```
{"store": {
  "name": "Sunshine Department Store",
  "address": "Wangfujing Street",
  "goods": [{
    "category": "Reference",
    "title": "Sayings of the Century",
    "author": "Nigel Rees",
    "price": 8.88
  },
  {
    "category": "Fiction",
    "title": "Sword of Honour",
    "author": "Evelyn Waugh",
    "price": 12.66
  }
  ],
  "bicycle": {
    "type": "GIANT OCR2600",
    "color": "White",
    "price": 276
  }
}}
```

In the following example, we will extract the store name, the store address, and the bicycle information from this file.

**Adding and linking the components**

**Procedure**

1. Create a new Job and add a `tFileInputJSON` component and a `tLogRow` component by typing their names in the design workspace or dropping them from the Palette.
2. Link the `tFileInputJSON` component to the `tLogRow` component using a `Row > Main` connection.

**Configuring the components**

**Procedure**

1. Double-click the `tFileInputJSON` component to open its Basic settings view.
2. Select **JsonPath without loop** from the **Read By** drop-down list. With this option, you need to specify the complete JSON path for each node of interest in the **JSONPath query** fields of the **Mapping** table.

3. Click the [...] button next to **Edit schema** to open the schema editor.

4. Click the [+ ] button to add five columns, **store_name**, **store_address**, **bicycle_type**, and **bicycle_color** of **String** type, and **bicycle_price** of **Double** type.

   Click **OK** to close the schema editor. In the pop-up dialog box, click **Yes** to propagate the schema to the subsequent component.

5. In the **Filename** field, specify the path to the JSON file that contains the data to be extracted. In this example, it is "E:/Store.json".

6. In the **Mapping** table, the **Column** fields are automatically filled with the schema columns you have defined.

   In the **JSONPath query** fields, enter the JSONPath query expressions between double quotation marks to specify the nodes that hold the desired data.
For the columns `store_name` and `store_address`, enter the JSONPath query expressions \\"$.store.name" and \\"$.store.address" relative to the nodes `name` and `address` respectively.

For the columns `bicycle_type`, `bicycle_color`, and `bicycle_price`, enter the JSONPath query expressions \\"$.store.goods.bicycle.type", \\"$.store.goods.bicycle.color", and \\"$.store.goods.bicycle.price" relative to the child nodes `type`, `color`, and `price` of the `bicycle` node respectively.

7. Double-click the `tLogRow` component to display its Basic settings view.

8. In the Mode area, select Table (print values in cells of a table) for better readability of the result.

## Executing the Job

**Procedure**

1. Press Ctrl+S to save the Job.
2. Press F6 to execute the Job.

```
[statistics] connecting to socket on port 374?
[statistics] connected

<table>
<thead>
<tr>
<th>store_name</th>
<th>store_address</th>
<th>bicycle_type</th>
<th>bicycle_color</th>
<th>bicycle_price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunshine Department Store</td>
<td>Nanjingfujing Street</td>
<td>GIANT OCR2600</td>
<td>White</td>
<td>276.0</td>
</tr>
</tbody>
</table>

[statistics] disconnected
```

As shown above, the store name, the store address, and the bicycle information are extracted from the source JSON data and displayed in a flat table on the console.

## Scenario 2: Extracting JSON data from a file using JSONPath

Based on Scenario 1: Extracting JSON data from a file using JSONPath without setting a loop node on page 1029, this scenario extracts data under the `book` array of the JSON file `Store.json` by specifying a loop node and the relative JSON path for each node of interest, and then displays the flat data extracted on the console.

### Procedure

1. In the Studio, open the Job used in Scenario 1: Extracting JSON data from a file using JSONPath without setting a loop node on page 1029 to display it in the design workspace.
2. Double-click the tFileInputJSON component to open its Basic settings view.

3. Select JsonPath from the Read By drop-down list.

4. In the Loop Json query field, enter the JSONPath query expression between double quotation marks to specify the node on which the loop is based. In this example, it is "$.store.goods.book[*]".

5. Click the [...] button next to Edit schema to open the schema editor.

Select the five columns added previously and click the [x] button to remove all of them.


Click OK to close the schema editor. In the pop-up dialog box, click Yes to propagate the schema to the subsequent component.

6. In the Json query fields of the Mapping table, enter the JSONPath query expressions between double quotation marks to specify the nodes that hold the desired data. In this example, enter the JSONPath query expressions "title", "category", "author", and "price" relative to the four child nodes of the book node respectively.

7. Press Ctrl+S to save the Job.
8. Press **F6** to execute the Job.

As shown above, the book information is extracted from the source JSON data and displayed in a flat table on the console.

**Scenario 3: Extracting JSON data from a file using XPath**

Based on **Scenario 1: Extracting JSON data from a file using JSONPath without setting a loop node** on page 1029, this scenario extracts the store name and the book information from the JSON file `Store.json` using XPath queries and displays the flat data extracted on the console.

**Procedure**

1. In the Studio, open the Job used in **Scenario 1: Extracting JSON data from a file using JSONPath without setting a loop node** on page 1029 to display it in the design workspace.
2. Double-click the **tFileInputJSON** component to open its **Basic settings** view.

3. Select **Xpath** from the **Read By** drop-down list.
4. Click the [...] button next to **Edit schema** to open the schema editor.
Select the five columns added previously and click the [x] button to remove all of them. Click the [+] button to add five columns, store_name, book_title, book_category, and book_author of String type, and book_price of Double type. Click OK to close the schema editor. In the pop-up dialog box, click Yes to propagate the schema to the subsequent component.

5. In the Loop XPath query field, enter the XPath query expression between double quotation marks to specify the node on which the loop is based. In this example, it is "/store/goods/book".

6. In the XPath query fields of the Mapping table, enter the XPath query expressions between double quotation marks to specify the nodes that hold the desired data.
   - For the column store_name, enter the XPath query ".//name" relative to the name node.

7. Press Ctrl+S to save the Job.
8. Press F6 to execute the Job.

As shown above, the store name and the book information are extracted from the source JSON data and displayed in a flat table on the console.

**Scenario 4: Extracting JSON data from a URL**

In this scenario, tFileInputJSON retrieves data of the friends node from the JSON file facebook.json on the Web that contains the data of a Facebook user and tExtractJSONFields extracts the data from the friends node for flat data output.
The JSON file `facebook.json` is deployed on the Tomcat server, specifically, located in the folder `<tomcat path>/webapps/docs`, and the content of the file is as follows:

```json
{
  "user": {
    "id": "9999912398",
    "name": "Kelly Clarkson",
    "friends": [
      {
        "name": "Tom Cruise",
        "id": "55555555555555",
        "likes": {
          "data": [
            {
              "category": "Movie",
              "name": "The Shawshank Redemption",
              "id": "103636093053996",
              "created_time": "2012-11-20T15:52:07+0000"
            },
            {
              "category": "Community",
              "name": "Positiveretribution",
              "id": "471389562899413",
              "created_time": "2012-12-16T21:13:26+0000"
            }
          ]
        }
      },
      {
        "name": "Tom Hanks",
        "id": "88888888888888",
        "likes": {
          "data": [
            {
              "category": "Journalist",
              "name": "Janelle Wang",
              "id": "136009823148851",
              "created_time": "2013-01-01T08:22:17+0000"
            },
            {
              "category": "Tv show",
              "name": "Now With Alex Wagner",
              "id": "305948749433410",
              "created_time": "2012-11-20T06:14:10+0000"
            }
          ]
        }
      }
    ]
  }
}
```

### Adding and linking the components

#### Procedure

1. Create a new Job and add a `tFileInputJSON` component, a `tExtractJSONFields` component, and two `tLogRow` components by typing their names in the design workspace or dropping them from the Palette.

2. Link the `tFileInputJSON` component to the first `tLogRow` component using a `Row > Main` connection.
3. Link the first tLogRow component to the tExtractJSONFields component using a Row > Main connection.

4. Link the tExtractJSONFields component to the second tLogRow component using a Row > Main connection.

Configuring the components

Procedure

1. Double-click the tFileInputJSON component to open its Basic settings view.

2. Select JsonPath without loop from the Read By drop-down list. Then select the Use Url check box and in the URL field displayed enter the URL of the file facebook.json from which the data will be retrieved. In this example, it is http://localhost:8080/docs/facebook.json.

3. Click the [...] button next to Edit schema and in the [Schema] dialog box define the schema by adding one column friends of String type.

4. In the Mapping table, enter the JSONPath query ".user.friends[*]" next to the friends column to retrieve the entire friends node from the source file.

5. Double-click tExtractJSONFields to open its Basic settings view.
6. Select Xpath from the Read By drop-down list.
7. In the Loop XPath query field, enter the XPath expression between double quotation marks to specify the node on which the loop is based. In this example, it is "/likes/data".
8. Click the [...] button next to Edit schema and in the [Schema] dialog box define the schema by adding five columns of String type, id, name, like_id, like_name, and like_category, which will hold the data of relevant nodes under the JSON field friends.

Click OK to close the dialog box and accept the propagation prompted by the pop-up dialog box.
9. In the XPath query fields of the Mapping table, type in the XPath query expressions between double quotation marks to specify the JSON nodes that hold the desired data. In this example:
   
   • ".//.//id" (querying the "/friends/id" node) for the column id,
   
   • ".//.//name" (querying the "/friends/name" node) for the column name,
   
   • "id" for the column like_id,
   
   • "name" for the column like_name, and
   
   • "category" for the column like_category.
10. Double-click the second tLogRow component to open its Basic settings view.
In the Mode area, select **Table (print values in cells of a table)** for better readability of the result.

### Executing the Job

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Click **F6** to execute the Job.

As shown above, the friends data in the JSON file specified using the URL is extracted and then the data from the node **friends** is extracted and displayed in a flat table.
tFileInputLDIF

Reads an LDIF file row by row to split them up into fields and sends the fields as defined in the schema to the next component using a Row connection.

tFileInputLDIF Standard properties

These properties are used to configure tFileInputLDIF running in the Standard Job framework.
The Standard tFileInputLDIF component belongs to the File family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

File Name

Name of the file and/or variable to be processed.
For further information about how to define and use a variable in a Job, see Talend Studio User Guide.

add operation as prefix when the entry is modify type

Select this check box to display the operation mode.

Value separator

Type in the separator required for parsing data in the given file. By default, the separator used is ",".

Die on error

Select this check box to stop the execution of the Job when an error occurs. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can collect the rows on error using a Row > Reject link.

Schema and Edit schema

A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- View schema: choose this option to view the schema only.
- Change to built-in property: choose this option to change the schema to Built-in for local changes.
- Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate
the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Advanced settings**

| Encoding | Select the encoding type from the list or select Custom and define it manually. This field is compulsory for DB data handling. |
|-----------------------------------------------|
| Use field options (for Base64 decode checked) | Select this check box to specify the Base64-encoded columns of the input flow. Once selected, this check box activates the Decode Base64 encoding values table to enable you to precise the columns to be decoded from Base64.  

**Note:**  
The data type of the columns to be handled by this check box is byte[] that you define in the input schema editor. |
| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

| Global Variables | NB_LINE: the number of rows processed. This is an After variable and it returns an integer.  
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Use this component to read full rows in a voluminous LDIF file. This component enables you to create a data flow, using a Row &gt; Main link, and to create a reject flow with a Row &gt; Reject link filtering the data which type does not match the defined type. For an example of usage, see Procedure on page 1094 from tFileInputXML.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided.</td>
</tr>
</tbody>
</table>
You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

**Related scenario**

For a related scenario, see *Scenario: Writing data from a database table into an LDIF file* on page 1132.
tFileInputMail

Reads the standard key data of a given MIME or MSG email file.

**tFileInputMail Standard properties**

These properties are used to configure tFileInputMail running in the Standard Job framework.

The Standard tFileInputMail component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>File Name</th>
<th>Specify the email file to read and extract data from.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description, it defines the number</td>
</tr>
<tr>
<td></td>
<td>of fields to be processed and passed on to the next</td>
</tr>
<tr>
<td></td>
<td>component. The schema is either <strong>Built-in</strong> or stored</td>
</tr>
<tr>
<td></td>
<td>remotely in the <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema.</td>
</tr>
<tr>
<td></td>
<td>If the current schema is of the <strong>Repository</strong> type,</td>
</tr>
<tr>
<td></td>
<td>three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the</td>
</tr>
<tr>
<td></td>
<td>schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option</td>
</tr>
<tr>
<td></td>
<td>to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option</td>
</tr>
<tr>
<td></td>
<td>to change the schema stored in the repository and</td>
</tr>
<tr>
<td></td>
<td>decide whether to propagate the changes to all the</td>
</tr>
<tr>
<td></td>
<td>Jobs upon completion. If you just want to propagate</td>
</tr>
<tr>
<td></td>
<td>the changes to the current Job, you can select <strong>No</strong></td>
</tr>
<tr>
<td></td>
<td>upon completion and choose this schema metadata</td>
</tr>
<tr>
<td></td>
<td>again in the <strong>Repository Content</strong> window.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: The schema</td>
<td>will be created and stored locally for this component</td>
</tr>
<tr>
<td></td>
<td>only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td><strong>Repository</strong>: The schema</td>
<td>already exists and is stored in the Repository, hence</td>
</tr>
<tr>
<td></td>
<td>can be reused in various projects and Job flowcharts.</td>
</tr>
<tr>
<td></td>
<td>Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>

**Mail type**

Select a type of email from the drop-down list, either **MIME** or **MSG**.

**Attachment export directory**

Specify the directory to which you want to export email attachments.

**Mail parts**

Specify what to extract from the defined MIME email file for each schema column.
• **Column**: The Column cells are automatically filled with the column name defined in the schema.

• **Mail part**: Type in the label of the header or body part to be extracted from the defined email.

• **Multi value**: Select the check box next to the name of columns made of multi-value fields.

• **Separator**: Enter a value separator for multi-value fields.

This table appears only when **MIME** is selected from the **Mail type** drop-down list.

**MSG Mail parts**

Specify what to extract from the defined MSG email file for each schema column.

• **Column**: The Column cells are automatically filled with the column name defined in the schema.

• **Mail part**: Click each cell and then select an email part to be extracted.

This table appears only when **MSG** is selected from the **Mail type** drop-down list.

**Die on error**

Select this check box to stop the execution of the Job when an error occurs. Clear the check box to skip the row on error and complete the process for error-free rows.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**Global Variables**

**EXPORTED_FILE_PATH**: the directory to export mail attachment. This is a Flow variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).
Usage

Usage rule
This component handles flow of data therefore it requires output. It is defined as an intermediary step.

Scenario: Extracting key fields from an email

This Java scenario describes a two-component Job that extracts some key standard fields and displays the values on the Run console.

Procedure

Procedure

1. Drop a tFileInputMail and a tLogRow component from the Palette to the design workspace.
2. Connect the two components together using a Main Row link.
3. Double-click tFileInputMail to display its Basic settings view and define the component properties.
4. Click the three-dot button next to the File Name field and browse to the mail file to be processed.
5. Set schema type to Built-in and click the three-dot button next to Edit schema to open a dialog box where you can define the schema including all columns you want to retrieve on your output.
6. Click the plus button in the dialog box to add as many columns as you want to include in the output flow. In this example, the schema has four columns: Date, Author, Object and Status.
7. Once the schema is defined, click OK to close the dialog box and propagate the schema into the Mail parts table.
8. Click the three-dot button next to Attachment export directory and browse to the directory in which you want to export email attachments, if any.
9. In the Mail part column of the Mail parts table, type in the actual header or body standard keys that will be used to retrieve the values to be displayed.

10. Select the Multi Value check box next to any of the standard keys if more than one value for the relative standard key is present in the input file.

11. If needed, define a separator for the different values of the relative standard key in the Separator field.

12. Double-click tLogRow to display its Basic settings view and define the component properties in order for the values to be separated by a carriage return. On Windows OS, type in `\n` between double quotes.

13. Save your Job and press F6 to execute it and display the output flow on the console.


[statistics] connecting to socket on port 3437
[statistics] connected
Wed. 04 Nov 2009 19:12:47 +0800
(CST) musicatcher@gmail.com | Talend multi value test | by
10.142.156.14 with SMTP id j14cs69293vff.
Wed. 04 Nov 2009 03:13:00 -0800 (PST)by 10.150.45.40
with SMTP id s40mr2413104ybe.260.1257333179981;
Wed. 04 Nov 2009 03:12:59 -0800 (PST)from
mail-gx0-f210.google.com (mail-gx0-f210.google.com
[209.85.217.210])
by mx.google.com with ESMTP id
33s11593716ywh.127.2009.11.04.03.12.58;
Wed. 04 Nov 2009 03:12:58 -0800 (PST)by
mail-gx0-f210.google.com with SMTP id 2mo6057662gxk4
for <musicatcher0@gmail.com>; Wed. 04 Nov 2009 03:12:58
-0800 (PST)by 10.150.75.12 with SMTP id
xl2mr2370266ybe.341.1257333175484;
Wed. 04 Nov 2009 03:12:55 -0800 (PST)from nsun
([219.237.242.224])
by mx.google.com with ESMTPS id
5sm364705ywd.23.2009.11.04.03.12.53
(version=SSLv3 cipher=RC4-MD5);
Wed. 04 Nov 2009 03:12:54 -0800 (PST)
[statistics] disconnected
Job tFileInputMail ended at 11:48 06/11/2009. [exit code=0]

Results

The header key values are extracted as defined in the Mail parts table. Mail reception date, author, subject and status are displayed on the console.
tFileInputMSDelimited

Reads the data structures (schemas) of a multi-structured delimited file and sends the fields as defined in the different schemas to the next components using Row connections.

**tFileInputMSDelimited Standard properties**

These properties are used to configure tFileInputMSDelimited running in the Standard Job framework. The Standard tFileInputMSDelimited component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Multi Schema Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <a href="#">Multi Schema Editor</a> helps to build and configure the data flow in a multi-structure delimited file to associate one schema per output. For more information, see The Multi Schema Editor on page 1048.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lists all the schemas you define in the <a href="#">Multi Schema Editor</a>, along with the related record type and the field separator that corresponds to every schema, if different field separators are used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Die on error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to stop the execution of the Job when an error occurs. Clear the check box to skip the row on error and complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Trim all column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to remove leading and trailing whitespaces from defined columns.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Validate date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to check the date format strictly against the input schema.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Advanced separator (for numbers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to modify the separators used for numbers:</td>
</tr>
<tr>
<td><strong>Thousands separator</strong>: define separators for thousands.</td>
</tr>
<tr>
<td><strong>Decimal separator</strong>: define separators for decimals.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

| Global Variables | NB_LINE: the number of rows processed. This is an After variable and it returns an integer. |
**Usage**

<table>
<thead>
<tr>
<th><strong>Usage rule</strong></th>
<th>Use this component to read multi-structured delimited files and separate fields contained in these files using a defined separator.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Limitation</strong></td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

**The Multi Schema Editor**

The [Multi Schema Editor] enables you to:

- set the path to the source file,
- define the source file properties,
- define data structure for each of the output schemas.

When you define data structure for each of the output schemas in the [Multi Schema Editor], column names in the different data structures automatically appear in the input schema lists of the components that come after tFileInputMSDelimited. However, you can still define data structures directly in the Basic settings view of each of these components.

The [Multi Schema Editor] also helps to declare the schema that should act as the source schema (primary key) from the incoming data to insure its unicity. The editor uses this mapping to associate all schemas processed in the delimited file to the source schema in the same file.

The editor opens with the first column, that usually holds the record type indicator, selected by default. However, once the editor is open, you can select the check box of any of the schema columns to define it as a primary key.

The below figure illustrates an example of the [Multi Schema Editor].
For detailed information about the usage of the Multi Schema Editor, see Scenario: Reading a multi structure delimited file on page 1049.

**Scenario: Reading a multi structure delimited file**

The following scenario creates a Java Job which aims at reading three schemas in a delimited file and displaying their data structure on the Run Job console.

The delimited file processed in this example looks like the following:
Dropping and linking components

Procedure

1. Drop a `tFileInputMSDelimited` component and three `tLogRow` components from the Palette onto the design workspace.

2. In the design workspace, right-click `tFileInputMSDelimited` and connect it to `tLogRow1`, `tLogRow2`, and `tLogRow3` using the `row_A_1`, `row_B_1`, and `row_C_1` links respectively.

Configuring the components

Procedure

1. Double-click `tFileInputMSDelimited` to open the Multi Schema Editor.
2. Click `Browse...` next to the **File name** field to locate the multi schema delimited file you need to process.

3. In the **File Settings** area:
   - Select from the list the encoding type the source file is encoded in. This setting is meant to ensure encoding consistency throughout all input and output files.
   - Select the field and row separators used in the source file.

   **Note:**
   Select the **Use Multiple Separator** check box and define the fields that follow accordingly if different field separators are used to separate schemas in the source file.

   A preview of the source file data displays automatically in the **Preview** panel.
Note:

Column 0 that usually holds the record type indicator is selected by default. However, you can select the check box of any of the other columns to define it as a primary key.

4. Click Fetch Codes to the right of the Preview panel to list the type of schema and records you have in the source file. In this scenario, the source file has three schema types (A, B, C). Click each schema type in the Fetch Codes panel to display its data structure below the Preview panel.

5. Click in the name cells and set column names for each of the selected schema.

In this scenario, column names read as the following:
- Schema A: Type, DiscName, Author, Date,
- Schema B: Type, SongName,
- Schema C: Type, LibraryName.
You need now to set the primary key from the incoming data to insure its unicity (DiscName in this scenario). To do that:

6. In the Fetch Codes panel, select the schema holding the column you want to set as the primary key (schema A in this scenario) to display its data structure.

7. Click in the Key cell that corresponds to the DiscName column and select the check box that appears.

8. Click anywhere in the editor and the false in the Key cell will become true.

You need now to declare the parent schema by which you want to group the other “children” schemas (DiscName in this scenario). To do that:

9. In the Fetch Codes panel, select schema B and click the right arrow button to move it to the right. Then, do the same with schema C.

Note:

The Cardinality field is not compulsory. It helps you to define the number (or range) of fields in “children” schemas attached to the parent schema. However, if you set the wrong number or range and try to execute the Job, an error message will display.

10. In the [Multi Schema Editor], click OK to validate all the changes you did and close the editor.

The three defined schemas along with the corresponding record types and field separators display automatically in the Basic settings view of tFileInputMSDelimited.
The three schemas you defined in the [Multi Schema Editor] are automatically passed to the three tLogRow components.

11. If needed, click the Edit schema button in the Basic settings view of each of the tLogRow components to view the input and output data structures you defined in the Multi Schema Editor or to modify them.

Saving and executing the Job

Procedure

1. Press Ctrl+S to save your Job.
2. Press F6 or click Run on the Run tab to execute the Job.

The multi schema delimited file is read row by row and the extracted fields are displayed on the Run Job console as defined in the [Multi Schema Editor].
Starting job pivot at 10:33 18/01/2010.

01|SOFT MUSIC ALBUM|RICHARDSON|15/12/2005
02|We Danced|SOFT MUSIC ALBUM
02|She's Everything|SOFT MUSIC ALBUM
02|Once in a Lifetime Love|SOFT MUSIC ALBUM
03|National Library|SOFT MUSIC ALBUM
01|COUNTRY MUSIC ALBUM|WHITE|02/01/2006
02|Fall Into Me|COUNTRY MUSIC ALBUM
02|Another Try|COUNTRY MUSIC ALBUM
02|Something About Her|COUNTRY MUSIC ALBUM

Job pivot ended at 10:33 18/01/2010. [exit code=0]
tFileInputMSPositional

Reads the data structures (schemas) of a multi-structured positional file and sends the fields as defined in the different schemas to the next components using Row connections.

**tFileInputMSPositional Standard properties**

These properties are used to configure tFileInputMSPositional running in the Standard Job framework. The Standard tFileInputMSPositional component belongs to the File family. The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property type</strong></td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>File Name</strong></td>
<td>Name of the file and/or the variable to be processed.</td>
</tr>
<tr>
<td></td>
<td>For further information about how to define and use a variable in a Job, see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Row separator</strong></td>
<td>String (ex: &quot;\n&quot; on Unix) to distinguish rows.</td>
</tr>
<tr>
<td><strong>Header Field Position</strong></td>
<td>Start-end position of the schema identifier.</td>
</tr>
<tr>
<td><strong>Records</strong></td>
<td>Schema: define as many schemas as needed.</td>
</tr>
<tr>
<td></td>
<td>Header value: value in the row that identifies a schema.</td>
</tr>
<tr>
<td></td>
<td>Pattern: string which represents the length of each column of the schema, separated by commas. Make sure the values defined in this field are relevant with the defined schema.</td>
</tr>
<tr>
<td></td>
<td>Reject incorrect row size: select the check boxes of the schemas where to reject incorrect row size.</td>
</tr>
<tr>
<td></td>
<td>Parent row: Select the parent row from the drop-down list. By default, it is &lt;Empty&gt;.</td>
</tr>
<tr>
<td></td>
<td>Parent key column: Type in the parent key column name. If the parent row is not &lt;Empty&gt;, this field must be filled with a column name of the parent row schema.</td>
</tr>
<tr>
<td></td>
<td>Key column: Type in the key column name.</td>
</tr>
<tr>
<td><strong>Skip from header</strong></td>
<td>Number of rows to be skipped in the beginning of file.</td>
</tr>
<tr>
<td><strong>Skip from footer</strong></td>
<td>Number of rows to be skipped at the end of the file.</td>
</tr>
<tr>
<td><strong>Limit</strong></td>
<td>Maximum number of rows to be processed. If Limit = 0, no row is read or processed.</td>
</tr>
</tbody>
</table>
### Die on parse error
Let the component die if an parsing error occurs.

### Die on unknown header type
Length values separated by commas, interpreted as a string between quotes. Make sure the values entered in this fields are consistent with the schema defined.

## Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Process long rows (needed for processing rows longer than 100,000 characters wide)</strong></td>
<td>Select this check box to process long rows (this is necessary to process rows longer than 100 000 characters).</td>
</tr>
</tbody>
</table>
| **Advanced separator (for numbers)** | Select this check box to modify the separators used for numbers: 
- **Thousands separator**: define separators for thousands. 
- **Decimal separator**: define separators for decimals. |
| **Trim all column** | Select this check box to remove leading and trailing whitespaces from defined columns. |
| **Validate date** | Select this check box to check the date format strictly against the input schema. |
| **Encoding** | Select the encoding type from the list or select **Custom** and define it manually. This field is compulsory for DB data handling. |
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

## Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong></td>
<td>the number of rows read by an input component or transferred to an output component. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_REJECTED</strong></td>
<td>the number of rows rejected. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_UNKNOWN_HEADER_TYPES</strong></td>
<td>the number of rows with unknown header type. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_PARSE_ERRORS</strong></td>
<td>the number of rows with parse errors. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

Use this component to read a multi schemas positional file and separate fields using a position separator value. You can also create a rejection flow using a Row > Reject link to filter the data which does not correspond to the type defined. For an example of how to use these two links, see Procedure on page 971.

Scenario: Reading data from a positional file

The following scenario reads data from a positional file, which contains two schemas. The positional file is shown below:

```
schema_1 (car_owner): schema_id; car_make; owner; age
schema_2 (car-insurance): schema_id; car_owner; age; car_insurance
1 bmw     John      45
1 bench   Mike      30
2 John    45 yes
2 Mike    50 No
```

Dropping the components

Procedure

1. Drop one tFileInputMSPositional and two tLogRow from the Palette to the design workspace.
2. Rename the two tLogRow components as car_owner and car_insurance.

Configuring the components

Procedure

1. Double-click the tFileInputMSPositional component to show its Basic settings view and define its properties.
2. In the **File name/Stream** field, type in the path to the input file. Also, you can click the [...] button to browse and choose the file.

3. In the **Header Field Position** field, enter the start-end position for the schema identifier in the input file, 0-1 in this case as the first character in each row is the schema identifier.

4. Click the [+] button twice to added two rows in the **Records** table.

5. Click the cell under the **Schema** column to show the [...] button.
   Click the [...] button to show the schema naming box.

6. Enter the schema name and click **OK**.
   The schema name appears in the cell and the schema editor opens.
7. Define the schema `car_owner`, which has four columns: `schema_id`, `car_make`, `owner` and `age`.

8. Repeat the steps to define the schema `car_insurance`, which has four columns: `schema_id`, `car_owner`, `age` and `car_insurance`.

9. Connect `tFileInputMSPositional` to the `car_owner` component with the `Row > car_owner` link, and the `car_insurance` component with the `Row > car_insurance` link.

10. In the `Header value` column, type in the schema identifier value for the schema, 1 for the schema `car_owner` and 2 for the schema `car_insurance` in this case.

11. In the `Pattern` column, type in the length of each field in the schema, the number of characters, number, etc in each field, 1,8,10,3 for the schema `car_owner` and 1,10,3,3 for the schema `car_insurance` in this case.

12. In the `Skip from header` field, type in the number of beginning rows to skip, 2 in this case as the two rows in the beginning just describes the two schemas, instead of the values.

13. Choose `Table (print values in cells of a table)` in the `Mode` area of the components `car_owner` and `car_insurance`.

**Executing the Job**

**Procedure**

1. Press `Ctrl+S` to save the Job.
2. Press `F6` or click `Run` on the `Run` tab to execute the Job.
The file is read row by row based on the length values defined in the **Pattern** field and output in two tables with different schemas.

<table>
<thead>
<tr>
<th>scheme_id</th>
<th>car_owner</th>
<th>age</th>
<th>car_insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>John</td>
<td>45</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>Mike</td>
<td>50</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>scheme_id</th>
<th>car_make</th>
<th>owner</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bmw</td>
<td>John</td>
<td>45</td>
</tr>
<tr>
<td>1</td>
<td>bench</td>
<td>Mike</td>
<td>30</td>
</tr>
</tbody>
</table>
tFileInputMSXML

Reads the data structures (schemas) of a multi-structured XML file and sends the fields as defined in the different schemas to the next components using Row connections.

**tFileInputMSXML Standard properties**

These properties are used to configure tFileInputMSXML running in the Standard Job framework. The Standard tFileInputMSXML component belongs to the File and the XML families. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File Name</strong></td>
<td>Name of the file and/or the variable to be processed. For further information about how to define and use a variable in a Job, see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Root XPath query</strong></td>
<td>The root of the XML tree, which the query is based on.</td>
</tr>
<tr>
<td><strong>Enable XPath in column &quot;Schema XPath loop&quot; but lose the order</strong></td>
<td>Select this check box if you want to define a XPath path in the Schema XPath loop field of the Outputs table while not keeping the order of the data shown in the source XML file.</td>
</tr>
<tr>
<td><strong>Outputs</strong></td>
<td>Schema: Define as many schemas as needed. Schema XPath loop: Enter the node of the XML tree or XPath path which the loop is based on. XPath Queries: Enter the fields to be extracted from the structured input. Create empty row: Select this check box if you want to create empty rows for the empty field(s) in the schema.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>Select this check box to stop the execution of the Job when an error occurs. Clear the check box to skip the row on error and complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trim all column</strong></td>
<td>Select this check box to remove leading and trailing whitespaces from defined columns.</td>
</tr>
<tr>
<td><strong>Validate date</strong></td>
<td>Select this check box to check the date format strictly against the input schema.</td>
</tr>
</tbody>
</table>
### tFileInputMSXML

<table>
<thead>
<tr>
<th>Ignore DTD file</th>
<th>Select this check box to ignore the DTD file indicated in the XML file being processed.</th>
</tr>
</thead>
</table>
| Generation mode | Select the appropriate generation mode according to your memory availability. The available modes are:  
- Slow and memory-consuming (Dom4j)  
  Note: This option allows you to use dom4j to process the XML files of high complexity.  
- Fast with low memory consumption (SAX) |
| Encoding        | Select the encoding type from the list or select CUSTOM and define it manually. This field is compulsory for DB data handling. |
| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

### Global Variables

**Global Variables**

- **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Scenario: Reading a multi-structure XML file

The following scenario describes a Job which reads a multi-structure XML file, extracts the desired fields and displays them on the console.

### Designing the Job

**Procedure**

1. Drop a tFileInputMSXML component from the Palette onto the design workspace and double-click the component to open its Basic settings view in the Component tab.
2. Browse to the XML file you want to process. In this example, it is `D:/Input/multischema_xml.xml`, which contains the following data:

```xml
<root>
  <toy>Cat</toy>
  <record>We Belong Together</record>
  <book>As You Like It</book>
  <book>All's Well That Ends Well</book>
  <record>When You Believe</record>
  <toy>Dog</toy>
</root>
```

3. In the **Root XPath query** field, enter the root of the XML tree, which the query will be based on. In this example, it is `/root`.

4. Select the **Enable XPath in column "Schema XPath loop" but lose the order** check box.

   In this example, to extract the desired fields, you need to define a XPath path in the **Schema XPath loop** field in the **Outputs** table for each output flow while not keeping the order of the data shown in the source XML file.

5. Click the plus button to add lines in the **Outputs** table where you can define the output schemas, `record` and `book` in this example.

6. In the **Outputs** table, click in the **Schema** cell and then click a three-dot button to display a dialog box where you can define the schema name.

   Enter a name for the output schema and click **OK** to close the dialog box.

7. The **tFileInputMSXML** schema editor appears.

   Define the schema according to your need.
8. Do the same to define the output schema record.

9. In the Schema XPath loop cell, enter the node of the XML tree, which the loop is based on. In this example, enter “/book” and “/record” respectively.

10. In the XPath Queries cell, enter the fields to be extracted from the structured XML input. In this example, enter the XPath query “.”.

11. In the design workspace, drop two tLogRow components from the Palette and connect tFileInputMSXML to tLogRow1 and tLogRow2 using the book and record links respectively. Rename the two tLogRow components as book and record respectively.

Saving and executing the Job

Procedure

1. Press Ctrl+S to save your Job.

2. Execute the Job by pressing F6 or clicking Run on the Run tab.

   The multi-structure XML file is read row by row and the extracted fields are displayed on the console. The first two fields are for the book schema, and the last two fields are for the record schema.
Starting job check at 17:58 07/05/2014.

[statistics] connecting to socket on port 3985
[statistics] connected
As You Like It
All's Well That Ends Well
We Belong Together
When You Believe
[statistics] disconnected
Job check ended at 17:58 07/05/2014. [exit code]
**tFileInputPositional**

Reads a positional file row by row to split them up into fields based on a given pattern and then sends the fields as defined in the schema to the next component.

**tFileInputPositional Standard properties**

These properties are used to configure tFileInputPositional running in the Standard Job framework. The Standard tFileInputPositional component belongs to the File family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored.</td>
</tr>
</tbody>
</table>

**File Name/Stream**

**File name**: Name and path of the file to be processed.

**Stream**: The data flow to be processed. The data must be added to the flow in order for tFileInputPositional to fetch these data via the corresponding representative variable.

This variable could be already pre-defined in your Studio or provided by the context or the components you are using along with this component, for example, the `INPUT_STREAM` variable of tFileFetch; otherwise, you could define it manually and use it according to the design of your Job, for example, using tJava or tJavaFlex.

In order to avoid the inconvenience of hand writing, you could select the variable of interest from the auto-completion list (Ctrl+Space) to fill the current field on condition that this variable has been properly defined.

Related topic to the available variables: see Talend Studio User Guide.

Related scenario to the input stream, see Scenario 2: Reading data from a remote file in streaming mode on page 1015.

**Row separator**

The separator used to identify the end of a row.

**Use byte length as the cardinality**

Select this check box to enable the support of double-byte character to this component. JDK 1.6 is required for this feature.

**Customize**

Select this check box to customize the data format of the positional file and define the table columns:

**Column**: Select the column you want to customize.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size</strong></td>
<td>Enter the column size.</td>
</tr>
<tr>
<td><strong>Padding char</strong></td>
<td>Enter, between double quotation marks, the padding character you need to remove from the field. A space by default.</td>
</tr>
<tr>
<td><strong>Alignment</strong></td>
<td>Select the appropriate alignment parameter.</td>
</tr>
<tr>
<td><strong>Pattern</strong></td>
<td>Length values separated by commas, interpreted as a string between quotes. Make sure the values entered in this field are consistent with the schema defined.</td>
</tr>
<tr>
<td><strong>Pattern Units</strong></td>
<td>The unit of the length values specified in the <strong>Pattern</strong> field.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Bytes</strong>: With this option selected, the length values in the <strong>Pattern</strong> field should be the count of bytes that represent symbols in the input file.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Symbols</strong>: With this option selected, the length values in the <strong>Pattern</strong> field should be the count of regular symbols, not including surrogate pairs.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Symbols (including rare)</strong>: With this option selected, the length values in the <strong>Pattern</strong> field should be the count of symbols, including surrogate pairs, and each surrogate pair counts as a single symbol. Considering the performance factor, it is not recommended to use this option when your input data consists of only regular symbols.</td>
</tr>
<tr>
<td><strong>Skip empty rows</strong></td>
<td>Select this check box to skip the empty rows.</td>
</tr>
<tr>
<td><strong>Uncompress as zip file</strong></td>
<td>Select this check box to uncompress the input file.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a <strong>Row &gt; Reject</strong> link.</td>
</tr>
<tr>
<td><strong>Header</strong></td>
<td>Enter the number of rows to be skipped in the beginning of file.</td>
</tr>
<tr>
<td><strong>Footer</strong></td>
<td>Number of rows to be skipped at the end of the file.</td>
</tr>
<tr>
<td><strong>Limit</strong></td>
<td>Maximum number of rows to be processed. If Limit = 0, no row is read or processed.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields. Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
</tbody>
</table>
• **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

This component must work with **tSetDynamicSchema** to leverage the dynamic schema feature.

| **Built-in** | The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide. |
| **Repository** | The schema already exists and is stored in the Repository, hence can be reused in various projects and Job flowcharts. Related topic: see Talend Studio User Guide. |

### Advanced settings

| **Needed to process rows longer than 100 000 characters** | Select this check box if the rows to be processed in the input file are longer than 100 000 characters. |
| **Advanced separator (for numbers)** | Select this check box to change the separator used for numbers. By default, the thousands separator is a comma (,) and the decimal separator is a period (.). **Thousands separator**: define separators for thousands. **Decimal separator**: define separators for decimals. |
| **Trim all column** | Select this check box to remove leading and trailing whitespaces from defined columns. |
| **Validate date** | Select this check box to check the date format strictly against the input schema. |
| **Encoding** | Select the encoding from the list or select **Custom** and define it manually. This field is compulsory for database data handling. |
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

### Global Variables

| **Global Variables** | **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer. **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions |
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | Use this component to read a file and separate fields using a position separator value. You can also create a rejection flow using a **Row > Reject** link to filter the data which does not correspond to the type defined. For an example of how to use these two links, see **Procedure** on page 971. |

### From Positional to XML file

The following scenario describes a two-component Job, which aims at reading data from an input file that contains contract numbers, customer references, and insurance numbers as shown below, and outputting the selected data (according to the data position) into an XML file.

<table>
<thead>
<tr>
<th>Contract</th>
<th>CustomerRef</th>
<th>InsuranceNr</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>8200</td>
<td>50330</td>
</tr>
<tr>
<td>00001</td>
<td>8201</td>
<td>50331</td>
</tr>
<tr>
<td>00002</td>
<td>8202</td>
<td>50332</td>
</tr>
<tr>
<td>00002</td>
<td>8203</td>
<td>50333</td>
</tr>
</tbody>
</table>

### Dropping and linking components

#### About this task

![Diagram](image)

#### Procedure

1. Drop a **tFileInputPositional** component from the **Palette** to the design workspace.
2. Drop a **tFileOutputXML** component as well. This file is meant to receive the references in a structured way.
3. Right-click the **tFileInputPositional** component and select **Row > Main**. Then drag it onto the **tFileOutputXML** component and release when the plug symbol shows up.
Configuring data input

Procedure

1. Double-click the `tFileInputPositional` component to show its **Basic settings** view and define its properties.

2. Define the Job Property type if needed. For this scenario, we use the built-in Property type.
   As opposed to the Repository, this means that the Property type is set for this station only.

3. Fill in a path to the input file in the **File Name** field. This field is mandatory.

4. Define the **Row separator** identifying the end of a row if needed, by default, a carriage return.

5. If required, select the **Use byte length as the cardinality** check box to enable the support of double-byte character.

6. Define the **Pattern** to delimit fields in a row. The pattern is a series of length values corresponding to the values of your input files. The values should be entered between quotes, and separated by a comma. Make sure the values you enter match the schema defined.

7. Fill in the **Header**, **Footer** and **Limit** fields according to your input file structure and your need. In this scenario, we only need to skip the first row when reading the input file. To do this, fill the **Header** field with `1` and leave the other fields as they are.

8. Next to **Schema**, select **Repository** if the input schema is stored in the Repository. In this use case, we use a **Built-In** input schema to define the data to pass on to the `tFileOutputXML` component.

9. You can load and/or edit the schema via the **Edit Schema** function. For this schema, define three columns, respectively **Contract**, **CustomerRef** and **InsuranceNr** matching the structure of the input file. Then, click **OK** to close the [Schema] dialog box and propagate the changes.
Configuring data output

Procedure

1. Double-click tFileOutputXML to show its Basic settings view.

2. Enter the XML output file path.

3. Define the row tag that will wrap each row of data, in this use case ContractRef.

4. Click the three-dot button next to Edit schema to view the data structure, and click Sync columns to retrieve the data structure from the input component if needed.

5. Switch to the Advanced settings tab view to define other settings for the XML output.
6. Click the plus button to add a line in the Root tags table, and enter a root tag (or more) to wrap the XML output structure, in this case ContractsList.

7. Define parameters in the Output format table if needed. For example, select the As attribute check box for a column if you want to use its name and value as an attribute for the parent XML element, clear the Use schema column name check box for a column to reuse the column label from the input schema as the tag label. In this use case, we keep all the default output format settings as they are.

8. To group output rows according to the contract number, select the Use dynamic grouping check box, add a line in the Group by table, select Contract from the Column list field, and enter an attribute for it in the Attribute label field.

9. Leave all the other parameters as they are.

**Saving and executing the Job**

**Procedure**

1. Press Ctrl+S to save your Job to ensure that all the configured parameters take effect.
2. Press F6 or click Run on the Run tab to execute the Job.

   The file is read row by row based on the length values defined in the Pattern field and output as an XML file as defined in the output settings. You can open it using any standard XML editor.
<?xml version="1.0" encoding="ISO-8859-15"?>

<ContractsList>
  <Contract Nr="00001">
    <ContractRef>
      <CustomerRef>8200</CustomerRef>
      <InsuranceNr>50330</InsuranceNr>
    </ContractRef>
  </Contract>

  <Contract Nr="00002">
    <ContractRef>
      <CustomerRef>8202</CustomerRef>
      <InsuranceNr>50332</InsuranceNr>
    </ContractRef>
  </Contract>

  <Contract Nr="00003">
    <ContractRef>
      <CustomerRef>8203</CustomerRef>
      <InsuranceNr>50333</InsuranceNr>
    </ContractRef>
  </Contract>

</ContractsList>
tFileInputProperties

Reads a text file row by row and separates the fields according to the model key = value.

**tFileInputProperties Standard properties**

These properties are used to configure tFileInputProperties running in the Standard Job framework.

The Standard tFileInputProperties component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository. For this component, the schema is read-only. It is made of two column, Key and Value, corresponding to the parameter name and the parameter value to be copied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>File format</td>
<td>Select from the list your file format, either: .properties or .ini.</td>
</tr>
<tr>
<td>.properties: data in the configuration file is written in two lines and structured according to the following way: key = value.</td>
<td></td>
</tr>
<tr>
<td>.ini: data in the configuration file is written in two lines and structured according to the following way: key = value and re-grouped in sections. Section Name: enter the section name on which the iteration is based.</td>
<td></td>
</tr>
<tr>
<td>File Name</td>
<td>Name and path to the file to be created and/or the variable to be used. For further information about how to define and use a variable, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Calculate MD5 Hash</td>
<td>Select this check box to verify that the file to be processed has been correctly downloaded.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Select the encoding type from the list or select Custom and define it manually. This field is compulsory for DB data handling.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>
Global Variables

| Global Variables | NB_LINE: the number of rows processed. This is an After variable and it returns an integer. ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | Use this component to read a text file and separate data according to the structure key = value. |

Scenario: Reading and matching the keys and the values of different .properties files and outputting the results in a glossary

This four-component Job reads two .properties files, one in French and the other in English. The data in the two input files is mapped to output a glossary matching the English and French terms.

The two input files used in this scenario hold localization strings for the tMysqlInput component in Talend Studio.
The glossary displays on the console listing three columns holding: the key name in the first column, the English term in the second, and the corresponding French term in the third.

**Dropping and linking the components**

**Procedure**

1. Drop the following components from the Palette onto the design workspace: `tFileInputProperties` (x2), `tMap`, and `tLogRow`.
2. Connect the component together using Row > Main links. The second properties file, FR, is used as a lookup flow.

![Diagram](image)

**Configuring the components**

**Procedure**

1. Double-click the first `tFileInputProperties` component to open its Basic settings view and define its properties.
2. In the **File Format** field, select your file format.

3. In the **File Name** field, click the three-dot button and browse to the input `.properties` file you want to use.

4. Do the same with the second **tFileInputProperties** and browse to the French `.properties` file this time.

5. Double-click the **tMap** component to open the **tMap** editor.
6. Select all columns from the *English_terms* table and drop them to the output table. Select the *key* column from the *English_terms* table and drop it to the *key* column in the *French_terms* table.

7. In the *glossary* table in the lower right corner of the *tMap* editor, rename the *value* field to *EN* because it will hold the values of the English file.

8. Click the plus button to add a line to the *glossary* table and rename it to *FR*.

9. In the *Length* field, set the maximum length to 255.

10. In the upper left corner of the *tMap* editor, select the *value* column in the *English_terms* table and drop it to the *FR* column in the *French_terms* table. When done, click OK to validate your changes and close the map editor and propagate the changes to the next component.

**Saving and executing the Job**

**Procedure**

1. Press Ctrl+S to save your Job.
2. Press F6 or click the Run button from the Run tab to execute it.
Starting job tFileInputProperties at 15:25 29/05/2009

<table>
<thead>
<tr>
<th>PORT.NAME</th>
<th>Port</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>HELP</td>
<td>org.talend.help.tMysqlInput</td>
<td>org.talend.help.tMysqlInput</td>
</tr>
<tr>
<td>STRING.QUOTE.NAME</td>
<td>String Quote</td>
<td>Séparateur de chaîne de caractère</td>
</tr>
<tr>
<td>DSTABLE.NAME</td>
<td>Table Name</td>
<td>Nom de table</td>
</tr>
<tr>
<td>QUERystore.NAME</td>
<td>Query Type</td>
<td>Type de requête</td>
</tr>
<tr>
<td>SQL_SYNTAX.NAME</td>
<td>Sql Syntax</td>
<td>Syntaxe SQL</td>
</tr>
<tr>
<td>TYPE.ITEM.PGSQL</td>
<td>PostgreSQL</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>TYPE.NAME</td>
<td>Database Driver</td>
<td>Pilote de base de données</td>
</tr>
<tr>
<td>TABLE.NAME</td>
<td>Table Name</td>
<td>Nom de table</td>
</tr>
<tr>
<td>ENCODING.NAME</td>
<td>Encoding</td>
<td>Encodage</td>
</tr>
<tr>
<td>QUERY.NAME</td>
<td>Query</td>
<td>Requête</td>
</tr>
<tr>
<td>DSDB-ODBC.INFO</td>
<td>Required for ODBC-like connection</td>
<td>Requis pour les connexions de type ODBC</td>
</tr>
</tbody>
</table>
tFileInputRaw

Reads all data in a raw file and sends it to a single output column for subsequent processing by another component.

**tFileInputRaw Standard properties**

These properties are used to configure tFileInputRaw running in the Standard Job framework.

The Standard tFileInputRaw component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file where the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td><strong>Filename</strong></td>
<td>The name of and path of the input file to be processed, which you can enter manually between double quotes or browse and select by clicking the [...] button.</td>
</tr>
</tbody>
</table>
| **Mode** | **Read the file as a string**: The content of the file is read as a string.  
**Read the file as a bytes array**: The content of the file is read as a bytes array.  
**Stream the file**: As soon as the first character is entered in the source file, it is read immediately. |
Encoding

If you are using the **Read the file as a string** mode, select the encoding type from the list or select **Custom** and define it manually.

Die on error

Select this check box to stop the execution of the Job when an error occurs. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can collect the rows on error using a **Row > Reject** link.

To catch the **FileNotFoundException**, you also need to select this check box.

Advanced settings

tStatCatcher Statistics

Select this check box to gather the processing metadata at the Job level as well as at each component level.

Global Variables

**FILENAME_PATH**: the path of the input file. This is an After variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.

Usage

**Usage rule**

Use this component to provide input data for Jobs that require a single column of data or that require a whole file to be read as a single column.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).
Related Scenario

For a related use case, see:

- Scenario: Uploading files to Dropbox on page 657
tFileInputRegex

Reads a file row by row to split them up into fields using regular expressions and sends the fields as defined in the schema to the next component.

Powerful feature which can replace number of other components of the File family. Requires some advanced knowledge on regular expression syntax.

tFileInputRegex Standard properties

These properties are used to configure tFileInputRegex running in the Standard Job framework.
The Standard tFileInputRegex component belongs to the File family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored.</td>
</tr>
</tbody>
</table>

File Name/Stream

File name: Name of the file and/or the variable to be processed
Stream: Data flow to be processed. The data must be added to the flow so that it can be collected by the tFileInputRegex via the INPUT_STREAM variable in the autocompletion list (Ctrl+Space)
For further information about how to define and use a variable in a Job, see Talend Studio User Guide.

Row separator

The separator used to identify the end of a row.

Regex

This field can contain multiple lines. Type in your regular expressions including the subpattern matching the fields to be extracted.

Note: Antislashes need to be doubled in regexp

Warning:

Regex syntax requires double quotes.

Header

Enter the number of rows to be skipped in the beginning of file.

Footer

Number of rows to be skipped at the end of the file.

Limit

Maximum number of rows to be processed. If Limit = 0, no row is read or processed.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ignore error message for the unmatched record</strong></td>
<td>Select this check box to avoid outputting error messages for records that do not match the specified regex. This check box is cleared by default.</td>
</tr>
</tbody>
</table>
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. |
| **Built-In**: | You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide. |
| **Repository**: | You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. |
| **Skip empty rows** | Select this check box to skip the empty rows. |
| **Die on error** | Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a **Row > Reject** link. |
| **Advanced settings** |  |
| **Encoding** | Select the encoding from the list or select **Custom** and define it manually. This field is compulsory for database data handling. In the Map/Reduce version of **tFileInputRegex**, you need to select the **Custom encoding** check box to display this list. |
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |
Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule | Use this component to read a file and separate fields contained in this file according to the defined Regex. You can also create a rejection flow using a Row > Reject link to filter the data which doesn’t correspond to the type defined. For an example of how to use these two links, see Procedure on page 971. |

Scenario: Regex to Positional file

The following scenario creates a two-component Job, reading data from an Input file using regular expression and outputting delimited data into a positional file.

Dropping and linking the components

Procedure

1. Drop a tFileInputRegex component from the Palette to the design workspace.
2. Drop a tFileOutputPositional component the same way.
3. Right-click on the tFileInputRegex component and select Row > Main. Drag this main row link onto the tFileOutputPositional component and release when the plug symbol displays.

Configuring the components

Procedure

1. Select the tFileInputRegex again so the Component view shows up, and define the properties:
2. The Job is built-in for this scenario. Hence, the Properties are set for this station only.

3. Fill in a path to the file in **File Name** field. This field is mandatory.

4. Define the **Row separator** identifying the end of a row.

5. Then define the **Regular expression** in order to delimit fields of a row, which are to be passed on to the next component. You can type in a regular expression using Java code, and on multiple lines if needed.

   **Warning:**

   *Regex syntax requires double quotes.*

6. In this expression, make sure you include all subpatterns matching the fields to be extracted.

7. In this scenario, ignore the header, footer and limit fields.

8. Select a local (Built-in) **Schema** to define the data to pass on to the **tFileOutputPositional** component.

9. You can load or create the schema through the **Edit Schema** function.

10. Then define the second component properties:

11. Enter the Positional file output path.

12. Enter the **Encoding** standard, the output file is encoded in. Note that, for the time being, the encoding consistency verification is not supported.
13. Select the **Schema** type. Click on **Sync columns** to automatically synchronize the schema with the Input file schema.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Now go to the **Run** tab, and click on **Run** to execute the Job.

The file is read row by row and split up into fields based on the **Regular Expression** definition. You can open it using any standard file editor.

<table>
<thead>
<tr>
<th>Fichier</th>
<th>Edition</th>
<th>Format</th>
<th>Affichage</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>82.120.169.93</td>
<td>20/Dec/2006:00:51:13</td>
<td>-0700/forum/profile.php</td>
<td></td>
<td></td>
</tr>
<tr>
<td>85.161.108.94</td>
<td>20/Dec/2006:00:51:13</td>
<td>-0700/forum</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
tFileInputXML

Reads an XML structured file row by row to split them up into fields and sends the fields as defined in the schema to the next component.

**tFileInputXML Standard properties**

These properties are used to configure tFileInputXML running in the Standard Job framework.

The Standard tFileInputXML component belongs to the File and the XML families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file where the properties are stored.</td>
</tr>
</tbody>
</table>

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-In** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

**File name/Stream**

**File name**: Name and path of the file to be processed.
Stream: The data flow to be processed. The data must be added to the flow in order for `tFileInputXML` to fetch these data via the corresponding representative variable.

This variable could be already pre-defined in your Studio or provided by the context or the components you are using along with this component, for example, the `INPUT_STREAM` variable of `tFileFetch`; otherwise, you could define it manually and use it according to the design of your Job, for example, using `tJava` or `tJavaFlex`.

In order to avoid the inconvenience of hand writing, you could select the variable of interest from the auto-completion list (`Ctrl+Space`) to fill the current field on condition that this variable has been properly defined.

Related topic to the available variables: see *Talend Studio User Guide*. Related scenario to the input stream, see Scenario 2: Reading data from a remote file in streaming mode on page 1015.

### Loop XPath query

Node of the tree, which the loop is based on.

### Mapping

**Column:** Columns to map. They reflect the schema as defined in the Schema type field.

**XPath Query:** Enter the fields to be extracted from the structured input.

**Get nodes:** Select this check box to recuperate the XML content of all current nodes specified in the XPath query list, or select the check box next to specific XML nodes to recuperate only the content of the selected nodes. These nodes are important when the output flow from this component needs to use the XML structure, for example, the Document data type.

For further information about the Document type, see *Talend Studio User Guide*.

**Note:**

The Get Nodes option functions in the DOM4j and SAX modes, although in SAX mode namespaces are not supported. For further information concerning the DOM4j and SAX modes, please see the properties noted in the Generation mode list of the Advanced Settings tab.

### Limit

Maximum number of rows to be processed. If Limit = 0, no row is read nor processed. If -1, all rows are read or processed.

### Die on error

Select the check box to stop the execution of the Job when an error occurs.

Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row > Reject link.
### Advanced settings

<table>
<thead>
<tr>
<th><strong>Ignore DTD file</strong></th>
<th>Select this check box to ignore the DTD file indicated in the XML file being processed.</th>
</tr>
</thead>
</table>
| **Advanced separator (for number)** | Select this check box to change the separator used for numbers. By default, the thousands separator is a comma (,) and the decimal separator is a period (.).  
  - **Thousands separator**: define the separators to use for thousands.  
  - **Decimal separator**: define the separators to use for decimals. |
| **Ignore the namespaces** | Select this check box to ignore name spaces.  
  - **Generate a temporary file**: click the three-dot button to browse to the XML temporary file and set its path in the field. |
| **Use Separator for mode Xerces** | Select this check box if you want to separate concatenated children node values.  
  - **Note:** This field can only be used if the selected **Generation mode** is Xerces.  
  The following field displays:  
  - **Field separator**: Define the delimiter to be used to separate the children node values. |
| **Encoding** | Select the encoding from the list or select **Custom** and define it manually. This field is compulsory for database data handling. |
| **Generation mode** | From the drop-down list select the generation mode for the XML file, according to the memory available and the desired speed:  
  - **Slow and memory-consuming (Dom4j)**  
    - **Note:** This option allows you to use dom4j to process the XML files of high complexity.  
  - **Memory-consuming (Xerces)**  
  - **Fast with low memory consumption (SAX)** |
| **Validate date** | Select this check box to check the date format strictly against the input schema. |
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

### Global Variables

| **Global Variables** | **error_message**: the error message generated by the component when an error occurs. This is an After |
variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

**NB_LINE:** The number of rows processed. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press \texttt{Ctrl + Space} to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

---

**Usage**

| Usage rule | tFileInputXML is for use as an entry component. It allows you to create a flow of XML data using a Row > Main link. You can also create a rejection flow using a Row > Reject link to filter the data which doesn't correspond to the type defined. For an example of how to use these two links, see Procedure on page 971. |

---

### Scenario 1: Reading and extracting data from an XML structure

This scenario describes a basic Job that reads a defined XML directory and extracts specific information and outputs it on the Run console via a \texttt{tLogRow} component.

---

### Procedure

1. Drop \texttt{tFileInputXML} and \texttt{tLogRow} from the Palette to the design workspace.
2. Connect both components together using a Main Row link.
3. Double-click \texttt{tFileInputXML} to open its Basic settings view and define the component properties.
4. As the street dir file used as input file has been previously defined in the Metadata area, select Repository as Property type. This way, the properties are automatically leveraged and the rest of the properties fields are filled in (apart from Schema). For more information regarding the metadata creation wizards, see Talend Studio User Guide.

5. Select the same way the relevant schema in the Repository metadata list. Edit schema if you want to make any change to the schema loaded.

6. The Filename shows the structured file to be used as input.

7. In Loop XPath query, change if needed the node of the structure where the loop is based.

8. On the Mapping table, fill the fields to be extracted and displayed in the output.

9. If the file size is consequent, fill in a Limit of rows to be read.

10. Enter the encoding if needed then double-click on tLogRow to define the separator character.

11. Save your Job and press F6 to execute it.

Results
Starting job XMLStreetFinder at 12:42 05/01/2007.
Paris|2eme arrondissement|Rue de la Paix
Paris|8eme arrondissement|Champs Elysees
New York City|Manhattan|Madison avenue
New York City|Brooklyn|Washington heights
Job XMLStreetFinder ended at 12:42 05/01/2007. jexit code

The fields defined in the input properties are extracted from the XML structure and displayed on the console.

**Scenario 2: Extracting erroneous XML data via a reject flow**

This Java scenario describes a three-component Job that reads an XML file and:

1. first, returns correct XML data in an output XML file,

2. and second, displays on the console erroneous XML data which type does not correspond to the defined one in the schema.
Procedure

1. Drop the following components from the Palette to the design workspace: tFileInputXML, tFileOutputXML, and tLogRow.
   Right-click tFileInputXML and select Row > Main in the contextual menu and then click tFileOutputXML to connect the components together.
   Right-click tFileInputXML and select Row > Reject in the contextual menu and then click tLogRow to connect the components together using a reject link.

2. Double-click tFileInputXML to display the Basic settings view and define the component properties.

3. In the Property Type list, select Repository and click the three-dot button next to the field to display the [Repository Content] dialog box where you can select the metadata relative to the input file if you have already stored it in the File xml node under the Metadata folder of the Repository tree view. The fields that follow are automatically filled with the fetched data. If not, select Built-in and fill in the fields that follow manually.
For more information about storing schema metadata in the Repository tree view, see *Talend Studio User Guide*.

4. In the **Schema Type** list, select **Repository** and click the three-dot button to open the dialog box where you can select the schema that describes the structure of the input file if you have already stored it in the **Repository** tree view. If not, select **Built-in** and click the three-dot button next to **Edit schema** to open a dialog box where you can define the schema manually.

The schema in this example consists of five columns: **id**, **CustomerName**, **CustomerAddress**, **idState** and **id2**.

5. Click the three-dot button next to the **Filename** field and browse to the XML file you want to process.

6. In the **Loop XPath query**, enter between inverted commas the path of the XML node on which to loop in order to retrieve data.

   In the **Mapping** table, **Column** is automatically populated with the defined schema.

   In the **XPath query** column, enter between inverted commas the node of the XML file that holds the data you want to extract from the corresponding column.

7. In the **Limit** field, enter the number of lines to be processed, the first 10 lines in this example.

8. Double-click **tFileOutputXML** to display its **Basic settings** view and define the component properties.

9. Click the three-dot button next to the **File Name** field and browse to the output XML file you want to collect data in, *customer_data.xml* in this example.

   In the **Row tag** field, enter between inverted commas the name you want to give to the tag that will hold the recuperated data.
Click **Edit schema** to display the schema dialog box and make sure that the schema matches that of the preceding component. If not, click **Sync columns** to retrieve the schema from the preceding component.

10. Double-click **tLogRow** to display its **Basic settings** view and define the component properties.

   Click **Edit schema** to open the schema dialog box and make sure that the schema matches that of the preceding component. If not, click **Sync columns** to retrieve the schema of the preceding component.

   In the **Mode** area, select the **Vertical** option.

11. Save your Job and press F6 to execute it.

**Results**

*Starting job tFileInputXML at 14:34 06/11/2009.*

```
[statistics] connecting to socket on port 3996
[statistics] connected
```

```
<table>
<thead>
<tr>
<th>#1. tLogRow_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
</tr>
<tr>
<td>id</td>
</tr>
<tr>
<td>CustomerName</td>
</tr>
<tr>
<td>CustomerAddress</td>
</tr>
<tr>
<td>idState</td>
</tr>
<tr>
<td>id2</td>
</tr>
<tr>
<td>errorCode</td>
</tr>
<tr>
<td>errorMessage</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>#2. tLogRow_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
</tr>
<tr>
<td>id</td>
</tr>
<tr>
<td>CustomerName</td>
</tr>
<tr>
<td>CustomerAddress</td>
</tr>
<tr>
<td>idState</td>
</tr>
<tr>
<td>id2</td>
</tr>
<tr>
<td>errorCode</td>
</tr>
<tr>
<td>errorMessage</td>
</tr>
</tbody>
</table>
```

*[statistics] disconnected
Job tFileInputXML ended at 14:35 06/11/2009. [exit code=0]*

The output file *customer_data.xml* holding the correct XML data is created in the defined path and erroneous XML data is displayed on the console of the Run view.
**tFileList**

Iterates a set of files or folders in a given directory based on a filemask pattern.

**tFileList Standard properties**

These properties are used to configure tFileList running in the Standard Job framework.

The Standard tFileList component belongs to the File and the Orchestration families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directory</td>
<td>Path to the directory where the files are stored.</td>
</tr>
<tr>
<td>FileList Type</td>
<td>Select the type of input you want to iterate on from the list:</td>
</tr>
<tr>
<td></td>
<td><strong>Files</strong> if the input is a set of files,</td>
</tr>
<tr>
<td></td>
<td><strong>Directories</strong> if the input is a set of directories,</td>
</tr>
<tr>
<td></td>
<td><strong>Both</strong> if the input is a set of the above two types.</td>
</tr>
<tr>
<td>Include subdirectories</td>
<td>Select this check box if the selected input source type includes sub-directories.</td>
</tr>
<tr>
<td>Case Sensitive</td>
<td>Set the case mode from the list to either create or not create case sensitive filter on filenames.</td>
</tr>
<tr>
<td>Generate Error if no file found</td>
<td>Select this check box to generate an error message if no files or directories are found.</td>
</tr>
<tr>
<td>Use Glob Expressions as Filemask</td>
<td>This check box is selected by default. It filters the results using a Global Expression (Glob Expressions).</td>
</tr>
<tr>
<td>Files</td>
<td>Click the plus button to add as many filter lines as needed:</td>
</tr>
<tr>
<td></td>
<td><strong>Filemask</strong>: in the added filter lines, type in a filename or a filemask using special characters or regular expressions.</td>
</tr>
<tr>
<td>Order by</td>
<td>The folders are listed first of all, then the files. You can choose to prioritise the folder and file order either:</td>
</tr>
<tr>
<td></td>
<td><strong>By default</strong>: alphabetical order, by folder then file;</td>
</tr>
<tr>
<td></td>
<td><strong>By file name</strong>: alphabetical order or reverse alphabetical order;</td>
</tr>
<tr>
<td></td>
<td><strong>By file size</strong>: smallest to largest or largest to smallest;</td>
</tr>
<tr>
<td></td>
<td><strong>By modified date</strong>: most recent to least recent or least recent to most recent.</td>
</tr>
</tbody>
</table>

**Note:**

If ordering by **file name**, in the event of identical file names then **modified date** takes precedence. If ordering by **file size**, in the event
of identical file sizes then file name takes precedence. If ordering by modified date, in the event of identical dates then file name takes precedence.

**Order action**

Select a sort order by clicking one of the following radio buttons:

- **ASC**: ascending order;
- **DESC**: descending order;

### Advanced settings

**Use Exclude Filemask**

Select this check box to enable Exclude Filemask field to exclude filtering condition based on file type:

- **Exclude Filemask**: Fill in the field with file types to be excluded from the Filemasks in the Basic settings view.

Note:

File types in this field should be quoted with double quotation marks and separated by comma.

**Format file path to slash(/) style (useful on Windows)**

Select this check box to format the file path to slash(/) style which is useful on Windows.

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

### Global Variables

**Global Variables**

- **CURRENT_FILE**: the current file name. This is a Flow variable and it returns a string.
- **CURRENT_FILEPATH**: the current file path. This is a Flow variable and it returns a string.
- **CURRENT_FILEEXTENSION**: the extension of the current file. This is a Flow variable and it returns a string.
- **CURRENT_FILEDIRECTORY**: the current file directory. This is a Flow variable and it returns a string.
- **NB_FILE**: the number of files iterated upon so far. This is a Flow variable and it returns an integer.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.
Scenario 1: Iterating on a file directory

The following scenario creates a three-component Job, which aims at listing files from a defined directory, reading each file by iteration, selecting delimited data and displaying the output in the Run log console.

Dropping and linking the components

Procedure
1. Drop the following components from the Palette to the design workspace: tFileList, tFileInputDelimited, and tLogRow.
2. Right-click the tFileList component, and pull an Iterate connection to the tFileInputDelimited component. Then pull a Main row from the tFileInputDelimited to the tLogRow component.

Configuring the components

Procedure
1. Double-click tFileList to display its Basic settings view and define its properties.
2. Browse to the **Directory** that holds the files you want to process. To display the path on the Job itself, use the label (**DIRECTORY**) that shows up when you put the pointer anywhere in the **Directory** field. Type in this label in the **Label Format** field you can find if you click the **View** tab in the **Basic settings** view.

3. In the **Basic settings** view and from the **FileList Type** list, select the source type you want to process, **Files** in this example.

4. In the **Case sensitive** list, select a case mode, **Yes** in this example to create case sensitive filter on file names.

5. Keep the **Use Glob Expressions as Filemask** check box selected if you want to use global expressions to filter files, and define a file mask in the **Filemask** field.

6. Double-click **tFileInputDelimited** to display its **Basic settings** view and set its properties.

7. Enter the **File Name** field using a variable containing the current filename path, as you filled in the **Basic settings** of **tFileList**. Press Ctrl+Space bar to access the autocomplete list of variables, and select the global variable `((String)globalMap.get("tFileList_1_CURRENT_FILEPATH"))`. This way, all files in the input directory can be processed.

8. Fill in all other fields as detailed in the **tFileInputDelimited** section. Related topic: **tFileInputDelimited** on page 1010.

9. Select the last component, **tLogRow**, to display its **Basic settings** view and fill in the separator to be used to distinguish field content displayed on the console. Related topic: **tLogRow** on page 1988.
**Executing the Job**

Press Ctrl + S to save your Job, and press F6 to run it.

Starting job test at 17:18 21/09/2009.

```
<table>
<thead>
<tr>
<th>id</th>
<th>CustomerName</th>
<th>CustomerAddress</th>
<th>idState</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Griffith Paving and Sealing</td>
<td>talent@apress91</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Bill's Dive Shop</td>
<td>511 Maple Ave. Apt. 1B</td>
<td>35</td>
</tr>
<tr>
<td>3</td>
<td>Childress Child Day Care</td>
<td>662 Lyons Circle</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Facelift Kitchen and Bath</td>
<td>220 Vine Ave.</td>
<td>41</td>
</tr>
<tr>
<td>5</td>
<td>Terrinni &amp; Son Auto and Truck</td>
<td>770 Enoor Rd.</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>Kermit the Pet Shop</td>
<td>1860 Ferkside In.</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>Tub's Furniture Store</td>
<td>807 Old Trail Rd.</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>Toggle &amp; Myers Ltd</td>
<td>618 Sheridan Rd.</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>Childress Child Day Care</td>
<td>788 Tennyson Ave.</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>Elle Hypnosis and Therapy Cent</td>
<td>2032 Northbrook Ct.</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Job test ended at 17:16 21/09/2009. (exit code=0)

The Job iterates on the defined directory, and reads all included files. Then delimited data is passed on to the last component which displays it on the console.

**Scenario 2: Finding duplicate files between two folders**

This scenario describes a Job that iterates on files in two folders, transforms the iteration results to data flows to obtain a list of filenames, and then picks up all duplicates from the list and shows them on the Run console, as a preparation step before merging the two folders, for example.
Dropping and linking the components

Procedure

1. From the Palette, drop two tFileList components, two tIterateToFlow components, two tFileOutputDelimited components, a tFileInputDelimited component, a tUniqRow component, and a tLogRow component onto the design workspace.

2. Link the first tFileList component to the first tIterateToFlow component using a Row > Iterate connection, and the connect the first tIterateToFlow component to the first tFileOutputDelimited component using a Row > Main connection to form the first subjob.

3. Link the second tFileList component to the second tIterateToFlow component using a Row > Iterate connection, and the connect the second tIterateToFlow component to the second tFileOutputDelimited component using a Row > Main connection to form the second subjob.

4. Link the tFileInputDelimited to the tUniqRow component using a Row > Main connection, and the tUniqRow component to the tLogRow component using a Row > Duplicates connection to form the third subjob.

5. Link the three subjobs using Trigger > On Subjob Ok connections so that they will be triggered one after another, and label the components to better identify their roles in the Job.

Configuring the components

Procedure

1. In the Basic settings view of the first tFileList component, fill the Directory field with the path to the first folder you want to read filenames from, E:/DataFiles/DI/images in this example, and leave the other settings as they are.
2. Double-click the first `tIterateToFlow` component to show its Basic settings view.

3. Double-click the [...] button next to Edit schema to open the [Schema] dialog box and define the schema of the text file the next component will write filenames to. When done, click OK to close the dialog box and propagate the schema to the next component.

   In this example, the schema contains only one column: Filename.
4. In **Value** field of the **Mapping** table, press **Ctrl+Space** to access the autocomplete list of variables, and select the global variable `((String)globalMap.get("tFileList_1_CURRENT_FILE"))` to read the name of each file in the input directory, which will be put into a data flow to pass to the next component.

5. In the **Basic settings** view of the first **tFileOutputDelimited** component, fill the **File Name** field with the path of the text file that will store the filenames from the incoming flow, `D:/temp/tempdata.csv` in this example. This completes the configuration of the first subjob.

6. Repeat the steps above to complete the configuration of the second subjob, but:
   - fill the **Directory** field in the **Basic settings** view of the second **tFileList** component with the other folder you want to read filenames from, `E:/DataFiles/DQ/images` in this example.
   - select the **Append** check box in the **Basic settings** view of the second **tFileOutputDelimited** component so that the filenames previously written to the text file will not be overwritten.

7. In the **Basic settings** view of the **tFileInputDelimited** component, fill the **File name/Stream** field with the path of the text file that stores the list of filenames, `D:/temp/tempdata.csv` in this example, and define the file schema, which contains only one column in this example, **Filename**.
8. In the **Basic settings** view of the **tUniqRow** component, select the **Key attribute** check box for the only column, **Filename** in this example.

9. In the **Basic settings** view of the **tLogRow** component, select the **Table (print values in cells of a table)** option for better display effect.

**Executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Click **Run** or press **F6** to run the Job.

   All the duplicate files between the selected folders are displayed on the console.
Results

For other scenarios using tFileList, see tFileCopy on page 983.
**tFileOutputARFF**

Writes an ARFF file that holds data organized according to the defined schema.

**tFileOutputARFF Standard properties**

These properties are used to configure tFileOutputARFF running in the Standard Job framework.

The Standard tFileOutputARFF component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>![Connection icon]</td>
<td>Click this icon to open a connection wizard and store the Excel file connection parameters you set in the component Basic settings view. For more information about setting up and storing file connection parameters, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>File name</td>
<td>Name or path to the output file and/or the variable to be used. For further information about how to define and use a variable in a Job, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Attribute Define</td>
<td>Displays the schema you defined in the [Edit schema] dialog box. <strong>Column</strong>: Name of the column. <strong>Type</strong>: Data type. <strong>Pattern</strong>: Enter the data model (pattern), if necessary.</td>
</tr>
<tr>
<td>Relation</td>
<td>Enter the name of the relation.</td>
</tr>
<tr>
<td>Append</td>
<td>Select this check box to add the new rows at the end of the file.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
</tbody>
</table>


- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th><strong>Built-in</strong></th>
<th>You can create the schema and store it locally for this component. Related topic: see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created and stored the schema in the Repository. You can reuse it in various projects and Job flowcharts. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

**Create directory if not exists**

This check box is selected by default. It creates a directory to hold the output table if it does not exist.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Don't generate empty file</strong></th>
<th>Select this check box if you do not want to generate empty files.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

**Global Variables**

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. <strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</th>
</tr>
</thead>
</table>
## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Use this component along with a Row link to collect data from another component and to re-write the data to an ARFF file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Connections

Outgoing links (from this component to another):
- **Row**: Main.
- **Trigger**: On Subjob Ok; On Subjob Error; Run if.

Incoming links (from one component to this one):
- **Row**: Main; Reject; Iterate.
- **Trigger**: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error; Synchronize; Parallelize.

For further information regarding connections, see Talend Studio User Guide.

### Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

### Related scenario

For **tFileOutputARFF** related scenario, see Scenario: Display the content of a ARFF file on page 1007.
tFileOutputDelimited

Outputs the input data to a delimited file according to the defined schema.

**tFileOutputDelimited Standard properties**

These properties are used to configure tFileOutputDelimited running in the Standard Job framework. The Standard tFileOutputDelimited component belongs to the File family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored.</td>
</tr>
</tbody>
</table>

**Use Output Stream**

Select this check box process the data flow of interest. Once you have selected it, the **Output Stream** field displays and you can type in the data flow of interest.

The data flow to be processed must be added to the flow in order for this component to fetch these data via the corresponding representative variable.

This variable could be already pre-defined in your Studio or provided by the context or the components you are using along with this component; otherwise, you could define it manually and use it according to the design of your Job, for example, using **tJava** or **tJavaFlex**.

In order to avoid the inconvenience of hand writing, you could select the variable of interest from the auto-completion list (Ctrl+Space) to fill the current field on condition that this variable has been properly defined.

For further information about how to use a stream, see **Scenario 2: Reading data from a remote file in streaming mode** on page 1015.

**File name**

Name or path to the output file and/or the variable to be used.

This field becomes unavailable once you have selected the **Use Output Stream** check box.

For further information about how to define and use a variable in a Job, see **Talend Studio User Guide**.

**Row Separator**

The separator used to identify the end of a row.

**Field Separator**

Enter character, string or regular expression to separate fields for the transferred data.
<table>
<thead>
<tr>
<th><strong>Append</strong></th>
<th>Select this check box to add the new rows at the end of the file.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Include Header</strong></td>
<td>Select this check box to include the column header to the file.</td>
</tr>
<tr>
<td><strong>Compress as zip file</strong></td>
<td>Select this check box to compress the output file in zip format.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields. Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Sync columns</strong></td>
<td>Click to synchronize the output file schema with the input file schema. The Sync function only displays once the <strong>Row</strong> connection is linked with the output component.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Advanced separator (for numbers)</strong></th>
<th>Select this check box to change the separator used for numbers. By default, the thousands separator is a comma (,) and the decimal separator is a period (.). <strong>Thousands separator</strong>: define separators for thousands. <strong>Decimal separator</strong>: define separators for decimals.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CSV options</strong></td>
<td>Select this check box to specify the following CSV parameters:</td>
</tr>
</tbody>
</table>
- **Escape char**: enter the escape character between double quotation marks. For example, you need to type in "\\" if the backslash character (\) is used as the escape character.

- **Text enclosure**: enter the enclosure character (only one character) between double quotation marks. For example, """ needs to be entered when double quotation marks (") are used as the enclosure character.

<table>
<thead>
<tr>
<th>Create directory if not exists</th>
<th>This check box is selected by default. It creates the directory that holds the output delimited file, if it does not already exist.</th>
</tr>
</thead>
</table>
| Split output in several files | In case of very big output files, select this check box to divide the output delimited file into several files.  
**Rows in each output file**: set the number of lines in each of the output files. |
| Custom the flush buffer size | Select this check box to define the number of lines to write before emptying the buffer.  
**Row Number**: set the number of lines to write. |
| Output in row mode | Select this check box to ensure atomicity of the flush so that each row of data can remain consistent as a set and incomplete rows of data are never written to a file.  
This check box is mostly useful when using this component in the multi-thread situation. |
| Encoding | Select the encoding from the list or select Custom and define it manually. This field is compulsory for database data handling. |
| Don’t generate empty file | Select this check box if you do not want to generate empty files. |
| Throw an error if the file already exist | Select this check box to throw an exception if the output file specified in the File Name field on the Basic settings view already exists.  
Clear this check box to overwrite the existing file. |
| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong></td>
<td>the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>FILE_NAME</strong></td>
<td>the name of the file being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>the error message generated by the component when an error occurs. This is an After variable.</td>
</tr>
</tbody>
</table>
variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Use this component to write a delimited file and separate fields using a field separator value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

### Scenario 1: Writing data in a delimited file

This scenario describes a three-component Job that extracts certain data from a file holding information about clients, customers, and then writes the extracted data in a delimited file.

In the following example, we have already stored the input schema under the Metadata node in the Repository tree view. For more information about storing schema metadata in the Repository, see Talend Studio User Guide.

### Dropping and linking components

#### Procedure

1. In the Repository tree view, expand Metadata and File delimited in succession and then browse to your input schema, customers, and drop it on the design workspace. A dialog box displays where you can select the component type you want to use.
2. Click `tFileInputDelimited` and then **OK** to close the dialog box. A `tFileInputDelimited` component holding the name of your input schema appears on the design workspace.

3. Drop a `tMap` component and a `tFileOutputDelimited` component from the **Palette** to the design workspace.

4. Link the components together using **Row > Main** connections.

**Configuring the components**

**Configuring the input component**

**Procedure**

1. Double-click `tFileInputDelimited` to open its **Basic settings** view. All its property fields are automatically filled in because you defined your input file locally.

   ![Basic settings view](image)

   - **Property Type**: *Repository*
   - **File name/Stream**: `D:\Input\customer.csv`
   - **Row Separator**: `\n`
   - **Field Separator**: `;`
   - **Header**: 6
   - **Footer**: 0
   - **Limit**: 0
   - **Built-In**: `DELIM:customer`
   - **Edit schema**: [ ]
   - **Uncompress as zip file**: [ ]
   - **Do on error**: [ ]

   2. If you do not define your input file locally in the **Repository** tree view, fill in the details manually after selecting **Built-in** in the **Property type** list.

   3. Click the [...] button next to the **File Name** field and browse to the input file, `customer.csv` in this example.

   **Warning:**
If the path of the file contains some accented characters, you will get an error message when executing your Job.

4. In the **Row Separators** and **Field Separators** fields, enter respectively \"\n\" and \",\" as line and field separators.

5. If needed, set the number of lines used as header and the number of lines used as footer in the corresponding fields and then set a limit for the number of processed rows.

   In this example, **Header** is set to 6 while **Footer** and **Limit** are not set.

6. In the **Schema** field, schema is automatically set to **Repository** and your schema is already defined since you have stored your input file locally for this example. Otherwise, select **Built-in** and click the [...] button next to **Edit Schema** to open the [Schema] dialog box where you can define the input schema, and then click **OK** to close the dialog box.

### Configuring the mapping component

**Procedure**

1. In the design workspace, double-click **tMap** to open its editor.

2. In the **tMap** editor, click on top of the panel to the right to open the **Add a new output table** dialog box.
3. Enter a name for the table you want to create, row2 in this example.
4. Click OK to validate your changes and close the dialog box.
5. In the table to the left, row1, select the first three lines (Id, CustomerName and CustomerAddress) and drop them to the table to the right.
6. In the Schema editor view situated in the lower left corner of the tMap editor, change the type of RegisterTime to String in the table to the right.

<table>
<thead>
<tr>
<th>Column</th>
<th>Key</th>
<th>Type</th>
<th>N.</th>
<th>Date Pattern</th>
<th>Length</th>
<th>Precision</th>
<th>Definition</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>idState</td>
<td></td>
<td>int</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>id</td>
<td></td>
<td>int</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RegTime</td>
<td></td>
<td>String</td>
<td></td>
<td></td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RegisterTime</td>
<td></td>
<td>String</td>
<td></td>
<td></td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum1</td>
<td></td>
<td>Float</td>
<td></td>
<td></td>
<td>10</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum2</td>
<td></td>
<td>Float</td>
<td></td>
<td></td>
<td>10</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. Click OK to save your changes and close the editor.

Configuring the output component

Procedure

1. In the design workspace, double-click tFileOutputDelimited to open its Basic settings view and define the component properties.

2. In the Property Type field, set the type to Built-in and fill in the fields that follow manually.
3. Click the [...] button next to the File Name field and browse to the output file you want to write data in, customerselection.txt in this example.
4. In the Row Separator and Field Separator fields, set "\n" and ";" respectively as row and field separators.
5. Select the Include Header check box if you want to output columns headers as well.
6. Click **Edit schema** to open the schema dialog box and verify if the recuperated schema corresponds to the input schema. If not, click **Sync Columns** to recuperate the schema from the preceding component.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Press **F6** or click **Run** on the **Run** tab to execute the Job.

```plaintext
customer selection.txt

<table>
<thead>
<tr>
<th>id</th>
<th>CustomerName</th>
<th>CustomerAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Griffith Paving and Sealcoating</td>
<td>talend@apres91</td>
</tr>
<tr>
<td>2</td>
<td>Bill's Dive Shop</td>
<td>511 Maple Ave. Apt. 1D</td>
</tr>
<tr>
<td>3</td>
<td>Childress Child Day Care</td>
<td>662 Lyons Circle</td>
</tr>
<tr>
<td>4</td>
<td>Facilt Kitchen and Bath</td>
<td>unknown</td>
</tr>
<tr>
<td>5</td>
<td>Terrinmi &amp; Son Auto and Truck</td>
<td>770 Exmoor Rd.</td>
</tr>
<tr>
<td>6</td>
<td>Kermit the Pet Shop</td>
<td>1060 Parkside Ln.</td>
</tr>
<tr>
<td>7</td>
<td>Tub's Furniture Store</td>
<td>307 Old Trail Rd.</td>
</tr>
<tr>
<td>8</td>
<td>Toggle &amp; Myerson Ltd</td>
<td>610 Sheridan Rd.</td>
</tr>
<tr>
<td>9</td>
<td>Childress Child Day Care</td>
<td>788 Tennyson Ave.</td>
</tr>
<tr>
<td>10</td>
<td>Elle Hypnosis and Therapy Cent</td>
<td>2C02 Northbrook Ct.</td>
</tr>
</tbody>
</table>
```

The three specified columns **id**, **CustomerName** and **CustomerAddress** are output in the defined output file.

**Scenario 2: Utilizing Output Stream to save filtered data to a local file**

Based on the preceding scenario, this scenario saves the filtered data to a local file using output stream.

**Dropping and linking components**

**Procedure**

1. Drop **tJava** from the **Palette** to the design workspace.
2. Connect **tJava** to **tFileInputDelimited** using a **Trigger > On Subjob OK** connection.

**Configuring the components**

**Procedure**

1. Double-click **tJava** to open its **Basic settings** view.

   ![tJava Basic settings](image)

   - In the **Code** area, type in the following command:
     ```java
default java.io.File("C:/myFolder").mkdirs();
globalMap.put("out_file", new
java.io.FileOutputStream("C:/myFolder/customerselection.txt",false));
```

   **Note:**
   In this scenario, the command we use in the **Code** area of **tJava** will create a new folder `C:/myFolder` where the output file `customerselection.txt` will be saved. You can customize the command in accordance with actual practice.

2. Double-click **tFileOutputDelimited** to open its **Basic settings** view.

   ![tFileOutputDelimited Basic settings](image)

   - Select the **Use Output Stream** check box to enable the **Output Stream** field in which you can define the output stream using command.
   - Fill in the **Output Stream** field with following command:
     ```java
     (java.io.OutputStream)globalMap.get("out_file")
     ```

   **Note:**
   You can customize the command in the **Output Stream** field by pressing **CTRL+SPACE** to select built-in command from the list or type in the command into the field manually in accordance with actual practice. In this scenario, the command we use in the **Output Stream** field will call the `java.io.OutputStream` class to output the filtered data stream to a local file which is defined in the **Code** area of **tJava** in this scenario.
5. Click **Sync columns** to retrieve the schema defined in the preceding component.
6. Leave rest of the components as they were in the previous scenario.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Press **F6** or click **Run** on the Run tab to execute the Job.

The three specified columns *Id, CustomerName* and *CustomerAddress* are output in the defined output file.

```
<table>
<thead>
<tr>
<th></th>
<th>Id: CustomerName: CustomerAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Griffith Paving and Sealcoating: talent@apres91</td>
</tr>
<tr>
<td>2</td>
<td>Bill's Dive Shop: 511 Maple Ave. Apt. 1D</td>
</tr>
<tr>
<td>3</td>
<td>Childress Child Day Care: 662 Lyons Circle</td>
</tr>
<tr>
<td>4</td>
<td>Facelift Kitchen and Bath: unknown</td>
</tr>
<tr>
<td>5</td>
<td>Terrini &amp; Son Auto and Truck: 770 Exmoor Rd.</td>
</tr>
<tr>
<td>6</td>
<td>Kermit the Pet Shop: 1060 Parkside Ln.</td>
</tr>
<tr>
<td>7</td>
<td>Tub's Furniture Store: 307 Old Trail Rd.</td>
</tr>
<tr>
<td>8</td>
<td>Toglie &amp; Myerson Ltd: 610 Sheridan Rd.</td>
</tr>
<tr>
<td>9</td>
<td>Childress Child Day Care: 788 Tennyson Ave.</td>
</tr>
<tr>
<td>10</td>
<td>Elle Hypnosis and Therapy Cen: 2032 Northbrook Ct.</td>
</tr>
</tbody>
</table>
```
**tFileOutputExcel**

Writes an MS Excel file with separated data values according to a defined schema.

### tFileOutputExcel Standard properties

These properties are used to configure tFileOutputExcel running in the Standard Job framework.

The Standard tFileOutputExcel component belongs to the File family.

The component in this framework is generally available.

#### Basic settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Write excel 2007 file format (xlsx)</strong></td>
<td>Select this check box to write the processed data into the .xlsx format of Excel 2007.</td>
</tr>
<tr>
<td><strong>Use Output Stream</strong></td>
<td>Select this check box to process the data flow of interest. Once you have selected it, the Output Stream field displays and you can type in the data flow of interest. The data flow to be processed must be added to the flow in order for this component to fetch these data via the corresponding representative variable. This variable could be already pre-defined in your Studio or provided by the context or the components you are using along with this component; otherwise, you could define it manually and use it according to the design of your Job, for example, using tJava or tJavaFlex. In order to avoid the inconvenience of writing manually, you could select the variable of interest from the auto-completion list (<strong>Ctrl</strong>+<strong>Space</strong>) to fill the current field on condition that this variable has been properly defined. For further information about how to use a stream, see <a href="#">Scenario 2: Reading data from a remote file in streaming mode</a> on page 1015.</td>
</tr>
<tr>
<td><strong>File name</strong></td>
<td>Name or path to the output file. This field becomes unavailable once you have selected the Use Output Stream check box. For further information about how to define and use a variable in a Job, see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
<tr>
<td><strong>Sheet name</strong></td>
<td>Name of the xsi sheet.</td>
</tr>
<tr>
<td><strong>Include header</strong></td>
<td>Select this check box to include a header row to the output file.</td>
</tr>
<tr>
<td><strong>Append existing file</strong></td>
<td>Select this check box to add the new lines at the end of the file. <strong>Append existing sheet</strong>: Select this check box to add the new lines at the end of the Excel sheet.</td>
</tr>
</tbody>
</table>
| Is absolute Y pos.          | Select this check box to add information in specified cells:
|                            | **First cell X**: cell position on the X-axis (X-coordinate or Abcissa).
|                            | **First cell Y**: cell position on the Y-axis (Y-coordinate).
|                            | **Keep existing cell format**: select this check box to retain the original layout and format of the cell you want to write into. |
| Font                      | Select in the list the font you want to use. |
| Define all columns auto size | Select this check box if you want the size of all your columns to be defined automatically. Otherwise, select the **Auto size** check boxes next to the column names you want their size to be defined automatically. |
| Schema and Edit Schema    | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**. |
|                           | Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available: |
|                           | • **View schema**: choose this option to view the schema only. |
|                           | • **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes. |
|                           | • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window. |
| Built-in                  | The schema will be created and stored locally for this component only. Related topic: see **Talend Studio User Guide**. |
| Repository                | The schema already exists and is stored in the Repository, hence can be reused in various projects and Job designs. Related topic: see **Talend Studio User Guide**. |
| Advanced settings         | **Create directory if not exists**: This check box is selected by default. This option creates the directory that will hold the output files if it does not already exist. |
|                           | **Custom the flush buffer size**: Available when **Write excel2007 file format** (xlsx) is selected in the **Basic settings** view. |
Select this check box to set the maximum number of rows in the **Row number** field that are allowed in the buffer.

**Advanced separator (for numbers)**  
Select this check box to modify the separators you want to use for numbers:  
- **Thousands separator**: define separators for thousands.  
- **Decimal separator**: define separators for decimals.

**Encoding**  
Select the encoding type from the list or select **Custom** and define it manually. This field is compulsory for DB data handling.

**Don’t generate empty file**  
Select the check box to avoid the generation of an empty file.

**Recalculate formula**  
Select this check box if you need to recalculate formula(s) in the specified Excel file.  
This check box appears only when you select all these three check boxes: **Write excel2007 file format(xlsx)**, **Append existing file**, and **Append existing sheet**.

**tStatCatcher Statistics**  
Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

---

### Global Variables

**Global Variables**  

- **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.

---

### Usage

**Usage rule**  
Use this component to write an MS Excel file with data passed on from other components using a **Row** link.

**Dynamic settings**  
Click the [*] button to add a row in the table and fill the **Code** field with a context variable to choose your HDFS connection dynamically from multiple connections.
planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

Related scenario

For tFileOutputExcel related scenario, see tSugarCRMInput on page 3783;

For scenario about the usage of Use Output Stream check box, see Scenario 2: Utilizing Output Stream to save filtered data to a local file on page 1117.
tFileOutputJSON

Receives data and rewrites it in a JSON structured data block in an output file.

**tFileOutputJSON Standard properties**

These properties are used to configure tFileOutputJSON running in the Standard Job framework.

The Standard tFileOutputJSON component belongs to the File family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>Name and path of the output file.</td>
</tr>
<tr>
<td>Generate an array json</td>
<td>Select this check box to generate an array JSON file.</td>
</tr>
<tr>
<td>Name of data block</td>
<td>Enter a name for the data block to be written, between double quotation marks. This field disappears when the Generate an array json check box is selected.</td>
</tr>
</tbody>
</table>
| Schema and Edit Schema         | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  - View schema: choose this option to view the schema only.  
  - Change to built-in property: choose this option to change the schema to Built-in for local changes.  
  - Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
once the Row connection is linked with the Output component.

**Advanced settings**

<table>
<thead>
<tr>
<th>Create directory if not exists</th>
<th>This check box is selected by default. This option creates the directory that will hold the output files if it does not already exist.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

**Usage**

| Usage rule | Use this component to rewrite received data in a JSON structured output file. |

**Scenario: Writing a JSON structured file**

This is a 2 component scenario in which a **tRowGenerator** component generates random data which a **tFileOutputJSON** component then writes to a JSON structured output file.
**Procedure**

1. Drop a **tRowGenerator** and a **tFileOutputJSON** component onto the workspace from the **Palette**.
2. Link the components using a **Row > Main** connection.
3. Double click **tRowGenerator** to define its **Basic Settings** properties in the **Component** view.

   ![tRowGenerator_1](image)

   **Basic settings**
   - **Schema**: Built-In
   - **RowGenerator Editor**: ...

4. Click `[...]` next to **Edit Schema** to display the corresponding dialog box and define the schema.

   ![Edit Schema](image)

5. Click `[+]` to add the number of columns desired.
6. Under **Columns** type in the column names.
7. Under **Type**, select the data type from the list.
8. Click **OK** to close the dialog box.
9. Click `[+]` next to **RowGenerator Editor** to open the corresponding dialog box.
10. Under **Functions**, select pre-defined functions for the columns, if required, or select [...] to set customized function parameters in the **Function parameters** tab.

11. Enter the number of rows to be generated in the corresponding field.

12. Click **OK** to close the dialog box.

13. Click **tFileOutputJSON** to set its **Basic Settings** properties in the **Component** view.

14. Click [...] to browse to where you want the output JSON file to be generated and enter the file name.

15. Enter a name for the data block to be generated in the corresponding field, between double quotation marks.

16. Select **Built-In** as the **Schema** type.

17. Click **Sync Columns** to retrieve the schema from the preceding component.

18. Press F6 to run the Job.
"person":[
    {
        "birthplace":"Manchester",
        "age":49,
        "name":"Carter"
    },
    {
        "birthplace":"Liverpool",
        "age":39,
        "name":"Clinton"
    },
    {
        "birthplace":"London",
        "age":53,
        "name":"Taylor"
    }
]

Results
The data from the input schema is written in a JSON structured data block in the output file.
tFileOutputLDIF

Writes or modifies an LDIF file with data separated in respective entries based on the schema defined, or else deletes content from an LDIF file.

tFileOutputLDIF outputs data to an LDIF type of file which can then be loaded into an LDAP directory.

**tFileOutputLDIF Standard properties**

These properties are used to configure tFileOutputLDIF running in the Standard Job framework.

The Standard tFileOutputLDIF component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>File Name</th>
<th>Specify the path to the LDIF output file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wrap</td>
<td>Specify the number of characters at which the line will be wrapped.</td>
</tr>
<tr>
<td>Change type</td>
<td>Select a changetype that defines the operation you want to perform on the entries in the output LDIF file.</td>
</tr>
<tr>
<td></td>
<td>• Add: the LDAP operation for adding the entry.</td>
</tr>
<tr>
<td></td>
<td>• Modify: the LDAP operation for modifying the entry.</td>
</tr>
<tr>
<td></td>
<td>• Delete: the LDAP operation for deleting the entry.</td>
</tr>
<tr>
<td></td>
<td>• Modrdn: the LDAP operation for modifying an entry’s RDN (Relative Distinguished Name).</td>
</tr>
<tr>
<td></td>
<td>• Default: the default LDAP operation.</td>
</tr>
</tbody>
</table>

| Multi-Values / Modify Detail | Specify the attributes for multi-value fields when Add or Default is selected from the Change type list or provide the detailed modification information when Modify is selected from the Change type list. |
|                            | • Column: The Column cells are automatically filled with the defined schema column names. |
|                            | • Operation: Select an operation to be performed on the corresponding field. This column is available only when Modify is selected from the Change type list. |
|                            | • MultiValue: Select the check box if the corresponding field is a multi-value field. |
|                            | • Separator: Specify the value separator in the corresponding multi-value field. |
|                            | • Binary: Select the check box if the corresponding field represents binary data. |
|                            | • Base64: Select the check box if the corresponding field should be base-64 encoded. The base-64 |
encoded data in the LDIF file is represented by the :: symbol.

This table is available only when **Add, Modify, or Default** is selected from the **Change type** list.

| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. |

| **Built-In** | You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**. |
| **Repository** | You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**. |

| **Sync columns** | Click to synchronize the output file schema with the input file schema. The Sync function only displays once the Row connection is linked with the Output component. |

| **Append** | Select this check box to add the new rows at the end of the file. |

| **Advanced settings** | | |
| **Enforce safe base 64 conversion** | Select this check box to enable the safe base-64 encoding. For more detailed information about the safe base-64 encoding, see [https://www.ietf.org/rfc/rfc2849.txt](https://www.ietf.org/rfc/rfc2849.txt). |
| **Create directory if not exists** | This check box is selected by default. It creates the directory that holds the output delimited file, if it does not already exist. |
| **Custom the flush buffer size** | Select this check box to specify the number of lines to write before emptying the buffer. |
### tFileOutputLDIF

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row number</strong></td>
<td>Type in the number of lines to write before emptying the buffer. This field is available only when the Custom the flush buffer size check box is selected.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select the encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td><strong>Don’t generate empty file</strong></td>
<td>Select this check box if you do not want to generate empty files.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

#### Global Variables

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong></td>
<td>the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

#### Usage

**Usage rule**

This component is used to write an LDIF file with data passed on from an input component using a Row > Main connection.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).
Scenario: Writing data from a database table into an LDIF file

This scenario describes a Job that loads data into a database table, and then extracts the data from the table and writing the data into a new LDIF file.

Adding and linking components

Procedure

1. Create a new Job and add the following components by typing their names in the design workspace or dropping them from the Palette: a tFixedFlowInput component, a tMysqlOutput component, a tMysqlInput component, and a tFileOutputLDIF component.

2. Link tFixedFlowInput to tMysqlOutput using a Row > Main connection.

3. Link tMysqlInput to tFileOutputLDIF using a Row > Main connection.

4. Link tFixedFlowInput to tMysqlInput using a Trigger > On Subjob Ok connection.

Configuring the components

Loading data into a database table

Procedure

1. Double-click tFixedFlowInput to open its Basic settings view.
2. Click the [...] button next to **Edit schema** and in the pop-up window define the schema by adding four columns: **dn**, **id_owners**, **registration**, and **make**, all of String type.

3. Click **OK** to close the schema editor and accept the propagation prompted by the pop-up dialog box.

4. In the **Mode** area, select **Use Inline Content(delimited file)**, and then in the **Content** field displayed, enter the following input data:

   24;24;5382 KC 94;Volkswagen
   32;32;9591 OE 79;Honda
   35;35;3129 VH 61;Volkswagen

5. Double-click **tMysqlOutput** to open its **Basic settings** view.

7. In the **Table** field, enter the name of the table into which the data will be written. In this example, it is **ldifdata**.

8. Select **Drop table if exists and create** from the **Action on table** drop-down list.

**Extracting data from the database table and writing it into an LDIF file**

**Procedure**

1. Double-click **tMysqlInput** to open its **Basic settings** view.
2. Fill in the **Host**, **Port**, **Database**, **Username**, and **Password** fields with your MySQL database connection details.

3. Click the [...] button next to **Edit schema** and in the pop-up window define the schema by adding four columns: `dn`, `id_owners`, `registration`, and `make`, all of String type.

4. In the **Table Name** field, enter the name of the table from which the data will be read. In this example, it is `ldifdata`.

5. Click the **Guess Query** button to fill in the **Query** field with the auto-generated query.

6. Double-click `tFileOutputLDIF` to open its **Basic settings** view.

![tFileOutputLDIF](image)

7. In the **File Name** field, browse to or enter the path to the LDIF file to be generated. In this example, it is `E:/out.ldif`.

8. Select the operation **Add** from the **Change type** list.

9. Click the **Sync columns** button to retrieve the schema from the preceding component.

### Saving and executing the Job

**Procedure**

1. Press **Ctrl+S** to save your Job.

2. Press **F6** or click **Run** on the **Run** tab to execute the Job.
The LDIF file created contains the data from the database table and the change type for the entries is set to **add**.
tFileOutputMSDelimited

Creates a complex multi-structured delimited file, using data structures (schemas) coming from several incoming Row flows.

**tFileOutputMSDelimited Standard properties**

These properties are used to configure tFileOutputMSDelimited running in the Standard Job framework.

The Standard tFileOutputMSDelimited component belongs to the File family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th><strong>File Name</strong></th>
<th>Name and path to the file to be created and/or the variable to be used. For further information about how to define and use a variable in a Job, see <a href="#">Talend Studio User Guide</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row Separator</strong></td>
<td>String (ex: &quot;\n&quot; on Unix) to distinguish rows.</td>
</tr>
<tr>
<td><strong>Field Separator</strong></td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td><strong>Use Multi Field Separators</strong></td>
<td>Select this check box to set a different field separator for each of the schemas using the Field separator field in the Schemas area.</td>
</tr>
</tbody>
</table>

### Schemas

The table gets automatically populated by schemas coming from the various incoming rows connected to tFileOutputMSDelimited. Fill out the dependency between the various schemas:

- **Parent row**: Type in the parent flow name (based on the Row name transferring the data).
- **Parent key column**: Type in the key column of the parent row.
- **Key column**: Type in the key column for the selected row.

### Advanced settings

- **Advanced separator (for numbers)**: Select this check box to modify the separators used for numbers:
  - **Thousands separator**: define separators for thousands.
  - **Decimal separator**: define separators for decimals.

- **CSV options**: Select this check box to take into account all parameters specific to CSV files, in particular Escape char and Text enclosure parameters.
### Create directory if not exists

This check box is selected by default. It creates the directory that holds the output delimited file, if it does not already exist.

### Encoding

Select the encoding from the list or select **Custom** and define it manually. This field is compulsory for DB data handling.

### Don’t generate empty file

Select this check box if you do not want to generate empty files.

### tStatCatcher Statistics

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>
**tFileOutputMSPositional**

Creates a complex multi-structured file, using data structures (schemas) coming from several incoming **Row** flows.

**tFileOutputMSPositional Standard properties**

These properties are used to configure tFileOutputMSPositional running in the Standard Job framework.

The Standard tFileOutputMSPositional component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File Name</strong></td>
<td>Name and path to the file to be created and/or variable to be used. For further information about how to define and use a variable in a Job, see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Row separator</strong></td>
<td>String (ex: &quot;\n&quot; on Unix) to distinguish rows.</td>
</tr>
<tr>
<td><strong>Schemas</strong></td>
<td>The table gets automatically populated by schemas coming from the various incoming rows connected to <strong>tFileOutputMSPositional</strong>. Fill out the dependency between the various schemas:</td>
</tr>
<tr>
<td></td>
<td><strong>Parent row</strong>: Type in the parent flow name (based on the Row name transferring the data).</td>
</tr>
<tr>
<td></td>
<td><strong>Parent key column</strong>: Type in the key column of the parent row.</td>
</tr>
<tr>
<td></td>
<td><strong>Key column</strong>: Type in the key column for the selected row.</td>
</tr>
<tr>
<td></td>
<td><strong>Pattern</strong>: Type in the pattern that positions the fields separator for each incoming row.</td>
</tr>
<tr>
<td></td>
<td><strong>Padding char</strong>: type in the padding character to be used</td>
</tr>
<tr>
<td></td>
<td><strong>Alignment</strong>: Select the relevant alignment parameter</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advanced separator (for numbers)</strong></td>
<td>Select this check box to modify the separators used for numbers:</td>
</tr>
<tr>
<td></td>
<td><strong>Thousands separator</strong>: define separators for thousands.</td>
</tr>
<tr>
<td></td>
<td><strong>Decimal separator</strong>: define separators for decimals.</td>
</tr>
<tr>
<td><strong>Create directory if not exists</strong></td>
<td>This check box is selected by default. It creates the directory that holds the output delimited file, if it does not already exist.</td>
</tr>
</tbody>
</table>
**tFileOutputMSPositional**

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for DB data handling.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is a Flow variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_REJECTED</strong>: the number of rows rejected. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_UNKNOWN_HEADER_TYPES</strong>: the number of rows with unknown header type. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_PARSE_ERRORS</strong>: the number of rows with parse errors. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>

**Usage**

| Usage rule | Use this component to write a multi-schema positional file and separate fields using a position separator value. |

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tFileOutputMSXML

Creates a complex multi-structured XML file, using data structures (schemas) coming from several incoming Row flows.

**tFileOutputMSXML Standard properties**

These properties are used to configure tFileOutputMSXML running in the Standard Job framework.

The Standard tFileOutputMSXML component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File Name</strong></td>
<td>Name and path to the file to be created and or the variable to be used. For further information about how to define and use a variable in a Job, see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Configure XML tree</strong></td>
<td>Opens the dedicated interface to help you set the XML mapping. For details about the interface, see Defining the MultiSchema XML tree on page 1142.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create directory only if not exists</td>
<td>This check box is selected by default. It creates the directory that holds the output delimited file, if it does not already exist.</td>
</tr>
</tbody>
</table>
| Advanced separator (for numbers) | Select this check box to modify the separators used for numbers:  
**Thousands separator:** define separators for thousands.  
**Decimal separator:** define separators for decimals. |
| Encoding                  | Select the encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling. |
| Don’t generate empty file | Select this check box if you do not want to generate empty files. |
| Trim the whitespace characters | Select this check box to remove leading and trailing whitespace from the columns. |
| Escape text               | Select this check box to escape special characters. |
| tStatCatcher Statistics   | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |
Global Variables

**NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Defining the MultiSchema XML tree

Double-click on the **tFileOutputMSXML** component to open the dedicated interface or click on the three-dot button on the **Basic settings** vertical tab of the **Component** tab.

To the left of the mapping interface, under **Linker source**, the drop-down list includes all the input schemas that should be added to the multi-schema output XML file (only if more than one input flow is connected to the **tFileOutputMSXML** component).
And under **Schema List**, are listed all columns retrieved from the input data flow in selection.

To the right of the interface, are expected all XML structures you want to create in the output XML file.

You can create manually or easily import the XML structures. Then map the input schema columns onto each element of the XML tree, respectively for each of the input schemas in selection under **Linker source**.

**Importing the XML tree**

The easiest and most common way to fill out the XML tree panel, is to import a well-formed XML file.

**Procedure**

1. Rename the **root tag** that displays by default on the **XML tree** panel, by clicking on it once.
2. Right-click on the root tag to display the contextual menu.
3. On the menu, select **Import XML tree**.
4. Browse to the file to import and click **OK**.
   - You can import an XML tree from files in XML, XSD and DTD formats.
   - When importing an XML tree structure from an XSD file, you can choose an element as the root of your XML tree.

   The **XML Tree** column is hence automatically filled out with the correct elements.
5. If you need to add or remove an element or sub-elements, right-click the relevant element of the tree to display the contextual menu.
6. Select **Delete** to remove the selection from the tree or select the relevant option among: **Add sub-element**, **Add attribute**, **Add namespace** to enrich the tree.

**Creating the XML tree manually**

If you don’t have any XML structure defined as yet, you can create it manually.

**Procedure**

1. Rename the **root tag** that displays by default on the **XML tree** panel, by clicking on it once.
2. Right-click on the root tag to display the contextual menu.
3. On the menu, select **Add sub-element** to create the first element of the structure.
4. If you need to add an attribute or a child element to any element or remove any element, right-click the left of the corresponding element name to display the contextual menu.
5. Right-click to the left of the element name to display the contextual menu.
6. On the menu, select the relevant option among: **Add sub-element**, **Add attribute**, **Add namespace** or **Delete**.

**Mapping XML data from multiple schema sources**

Once your XML tree is ready, select the first input schema that you want to map.

You can map each input column with the relevant XML tree element or sub-element to fill out the **Related Column**.
Procedure

1. Click on one of the **Schema column name**.
2. Drag it onto the relevant sub-element to the right.
3. Release the mouse button to implement the actual mapping.

A light blue link displays that illustrates this mapping. If available, use the **Auto-Map** button, located to the bottom left of the interface, to carry out this operation automatically.

4. If you need to disconnect any mapping on any element of the XML tree, select the element and right-click to the left of the element name to display the contextual menu.
5. Select **Disconnect link**.
   The light blue link disappears.

**Defining the node status**

Defining the XML tree and mapping the data is not sufficient. You also need to define the loop elements for each of the source in selection and if required the group element.

**Define a loop element**

The loop element allows you to define the iterating object. Generally the Loop element is also the row generator.

**About this task**

To define an element as loop element:

**Procedure**

1. Select the relevant element on the XML tree.
2. Right-click to the left of the element name to display the contextual menu.
3. Select **Set as Loop Element**.

**Results**

The **Node Status** column shows the newly added status.

There can only be one loop element at a time.
**Define a group element**

The group element is optional, it represents a constant element where the groupby operation can be performed. A group element can be defined only if a loop element was defined before.

**About this task**

When using a group element, the rows should sorted, in order to be able to group by the selected node.

To define an element as group element:

**Procedure**

1. Select the relevant element on the XML tree.
2. Right-click to the left of the element name to display the contextual menu.
3. Select **Set as Group Element**.

**Results**

The **Node Status** column shows the newly added status and any group status required are automatically defined, if needed.

Click **OK** once the mapping is complete to validate the definition and continue the job configuration where needed.

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tFileOutputPositional

Writes a file row by row according to the length and the format of the fields or columns in a row.

**tFileOutputPositional Standard properties**

These properties are used to configure tFileOutputPositional running in the Standard Job framework.

The Standard tFileOutputPositional component belongs to the File family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored.</td>
</tr>
</tbody>
</table>

**Use Output Stream**

Select this check box to process the data flow of interest. Once you have selected it, the **Output Stream** field displays and you can type in the data flow of interest.

The data flow to be processed must be added to the flow in order for this component to fetch these data via the corresponding representative variable.

This variable could be already pre-defined in your Studio or provided by the context or the components you are using along with this component; otherwise, you could define it manually and use it according to the design of your Job, for example, using **tJava** or **tJavaFlex**.

In order to avoid the inconvenience of hand writing, you could select the variable of interest from the auto-completion list (**Ctrl+Space**) to fill the current field on condition that this variable has been properly defined.

For further information about how to use a stream, see **Scenario 2: Reading data from a remote file in streaming mode** on page 1015.

**File Name**

Name or path to the file to be processed and or the variable to be used.

This field becomes unavailable once you have selected the **Use Output Stream** check box.

For further information about how to define and use a variable in a Job, see **Talend Studio User Guide**.

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields.
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.

**Row separator**

The separator used to identify the end of a row.

**Append**

Select this check box to add the new rows at the end of the file.

**Include header**

Select this check box to include the column header to the file.

**Compress as zip file**

Select this check box to compress the output file in zip format.

**Formats**

Customize the positional file data format and fill in the columns in the Formats table.

- **Column**: Select the column you want to customize.
- **Size**: Enter the column size.
- **Padding char**: Type in between quotes the padding characters used. A space by default.
- **Alignment**: Select the appropriate alignment parameter.
- **Keep**: If the data in the column or in the field are too long, select the part you want to keep.

**Advanced settings**

**Advanced separator (for numbers)**

Select this check box to change the separator used for numbers. By default, the thousands separator is a comma (,) and the decimal separator is a period (.)

- **Thousands separator**: define separators for thousands.
- **Decimal separator**: define separators for decimals.
Use byte length as the cardinality

Select this check box to add support of double-byte character to this component. JDK 1.6 is required for this feature.

Create directory if not exists

This check box is selected by default. It creates a directory to hold the output table if it does not exist.

Custom the flush buffer size

Select this check box to define the number of lines to write before emptying the buffer.

**Row Number:** set the number of lines to write.

Output in row mode

Writes in row mode.

Encoding

Select the encoding from the list or select **Custom** and define it manually. This field is compulsory for database data handling.

Don't generate empty file

Select this check box if you do not want to generate empty files.

tStatCatcher Statistics

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

Global Variables

**Global Variables**

**NB_LINE:** the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

**ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**

Use this component to read a file and separate the fields using the specified separator.

**Dynamic settings**

Click the [+ ] button to add a row in the table and fill the **Code** field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working...
in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenario

For a related scenario, see Scenario: Regex to Positional file on page 1086.

For scenario about the usage of Use Output Stream check box, see Scenario 2: Utilizing Output Stream to save filtered data to a local file on page 1117.
tFileOutputProperties

Writes a configuration file, of the type `.ini` or `.properties`, containing text data organized according to the model key = value.

**tFileOutputProperties Standard properties**

These properties are used to configure tFileOutputProperties running in the Standard Job framework. The Standard tFileOutputProperties component belongs to the File family. The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository. For this component, the schema is read-only. It is made of two column, Key and Value, corresponding to the parameter name and the parameter value to be copied.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File format</strong></td>
<td>Select from the list file format: either <code>.properties</code> or <code>.ini</code>.</td>
</tr>
<tr>
<td><strong>.properties:</strong> data in the configuration file is written in two lines and structured according to the following way: key = value.</td>
<td></td>
</tr>
<tr>
<td><strong>.ini:</strong> data in the configuration file is written in two lines and structured according to the following way: key = value and re-grouped in sections. <strong>Section Name:</strong> enter the section name on which the iteration is based.</td>
<td></td>
</tr>
<tr>
<td><strong>File Name</strong></td>
<td>Name or path to the file to be processed and/or the variable to be used. For further information about how to define and use a variable in a Job, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for DB data handling.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>
Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule | Use this component to write files where data is organized according to the structure key = value. |

Related scenarios

For a related scenario, see Scenario: Reading and matching the keys and the values of different .properties files and outputting the results in a glossary on page 1076 of tFileInputProperties on page 1075.
tFileOutputRaw

Provides data coming from another component, in the form of a single column of output data.

**tFileOutputRaw Standard properties**

These properties are used to configure tFileOutputRaw running in the Standard Job framework.

The Standard tFileOutputRaw component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the [Repository].

Click [Edit schema] to make changes to the schema. If the current schema is of the [Repository] type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Built-in</th>
<th>No property data stored centrally.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>Select the repository file where the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Filename**

The name of and path to the output file to be processed, which you can enter manually between double quotes or browse and select by clicking the [...] button.

**Encoding**

If the output is a string, select the encoding type from the list or select **Custom** and define it manually.

**Die on error**

Select this check box to stop the execution of the Job when an error occurs. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can collect the rows on error using a [Row > Reject] link.

To catch the `FileNotFoundException`, you also need to select this check box.
### Advanced settings

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>FILENAME_PATH: the path of the input file. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

| Usage rule | Use the tFileOutputRaw component to receive data coming from a data source that provides its data in a single column. |

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |
tFileOutputXML

Writes an XML file with separated data values according to a defined schema.

**tFileOutputXML Standard properties**

These properties are used to configure tFileOutputXML running in the Standard Job framework.

The Standard tFileOutputXML component belongs to the File and the XML families.

The component in this framework is generally available.

**Basic settings**

| **File name** | Name or path to the output file and/or the variable to be used.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Related topic: see Defining variables from the Component view section in <em>Talend Studio User Guide</em></td>
</tr>
</tbody>
</table>

| **Incoming record is a document** | Select this check box if the data from the preceding component is in XML format.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When this check box is selected, a Column list appears allowing you to select a Document type column of the schema that holds the data, and the Row tag field disappears.</td>
</tr>
<tr>
<td></td>
<td>When this check box is selected, in the Advanced settings view, only the check boxes Create directory if not exists, Don’t generate empty file, Trim data, tStatCatcher Statistics and the list Encoding are available.</td>
</tr>
</tbody>
</table>

| **Row tag** | Specify the tag that will wrap data and structure per row. |

<table>
<thead>
<tr>
<th><strong>Schema and Edit Schema</strong></th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
</tbody>
</table>
Built-In: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

**Sync columns**

Click to synchronize the output file schema with the input file schema. The Sync function only displays once the Row connection is linked with the input component.

### Advanced settings

<table>
<thead>
<tr>
<th>Split output in several files</th>
<th>If the output is big, you can split the output into several files, each containing the specified number of rows.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rows in each output file:</strong></td>
<td>Specify the number of rows in each output file.</td>
</tr>
<tr>
<td>Create directory if not exists</td>
<td>This check box is selected by default. It creates a directory to hold the output XML files if required.</td>
</tr>
<tr>
<td>Root tags</td>
<td>Specify one or more root tags to wrap the whole output file structure and data. The default root tag is <em>root</em>.</td>
</tr>
<tr>
<td><strong>Output format</strong></td>
<td>Define the output format.</td>
</tr>
<tr>
<td>• <strong>Column</strong>: The columns retrieved from the input schema.</td>
<td></td>
</tr>
<tr>
<td>• <strong>As attribute</strong>: select check box for the column(s) you want to use as attribute(s) of the parent element in the XML output.</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>If the same column is selected in both the <strong>Output format</strong> table as an attribute and in the <strong>Use dynamic grouping</strong> setting as the criterion for dynamic grouping, only the dynamic group setting will take effect for that column.</td>
</tr>
<tr>
<td><strong>Use schema column name</strong>: By default, this check box is selected for all columns so that the column labels from the input schema are used as data wrapping tags. If you want to use a different tag than from the input schema for any column, clear this check box for that column and specify a tag label between quotation marks in the <strong>Label</strong> field.</td>
<td></td>
</tr>
<tr>
<td><strong>Use dynamic grouping</strong></td>
<td>Select this check box if you want to dynamically group the output columns. Click the plus button to add one or more grouping criteria in the <strong>Group by</strong> table.</td>
</tr>
<tr>
<td><strong>Column</strong>: Select a column you want to use as a wrapping element for the grouped output rows.</td>
<td></td>
</tr>
<tr>
<td><strong>Attribute label</strong>: Enter an attribute label for the group wrapping element, between quotation marks.</td>
<td></td>
</tr>
</tbody>
</table>
### Custom the flush buffer size
Select this check box to define the number of rows to buffer before the data is written into the target file and the buffer is emptied.

**Row Number**: Specify the number of rows to buffer.

### Advanced separator (for numbers)
Select this check box to change the separator used for numbers. By default, the thousands separator is a comma (,) and the decimal separator is a period (.).

**Thousands separator**: define separators for thousands.

**Decimal separator**: define separators for decimals.

### Encoding
Select the encoding from the list or select **Custom** and define it manually. This field is compulsory for database data handling.

### Don’t generate empty file
Select the check box to avoid the generation of an empty file.

### Trim data
Select this check box to remove the spaces at the beginning and at the end of the text, and merge multiple consecutive spaces into one within the text.

### tStatCatcher Statistics
Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MESSAGE</strong></td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td><strong>NB_LINE</strong></td>
<td>the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage rule</strong></td>
<td>Use this component to write an XML file with data passed on from other components using a <strong>Row</strong> link.</td>
</tr>
</tbody>
</table>
Related scenarios

For related scenarios using `tFileOutputXML`, see *From Positional to XML file* on page 1070 and *Scenario 2: Using a SOAP message from an XML file to get country name information and saving the information to an XML file* on page 3565.
tFileProperties

Creates a single row flow that displays the main properties of the processed file.

**tFileProperties Standard properties**

These properties are used to configure tFileProperties running in the Standard Job framework.

The Standard tFileProperties component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

| **Schema and Edit schema** | A schema is a row description, it defines the fields to be processed and passed on to the next component. The schema of this component is read-only. It describes the main properties of the specified file. You can click the [...] button next to **Edit schema** to view the predefined schema which contains the following fields:
| | • **abs_path**: the absolute path of the file.
| | • **dirname**: the directory of the file.
| | • **basename**: the name of the file.
| | • **mode_string**: the access mode of the file, r and w for read and write permissions respectively.
| | • **size**: the file size in bytes.
| | • **mtime**: the timestamp indicating when the file was last modified, in milliseconds that have elapsed since the Unix epoch (00:00:00 UTC, Jan 1, 1970).
| | • **mtime_string**: the date and time the file was last modified.
| **File** | Name or path to the file to be processed and/or the variable to be used.
| | For further information about how to define and use a variable in a Job, see **Talend Studio User Guide**.
| **Calculate MD5 Hash** | Select this check box to check the MD5 of the downloaded file.

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |
Global Variables

Global Variables

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as standalone component.</th>
</tr>
</thead>
</table>

Connections

Outgoing links (from this component to another): **Row**: Main; Iterate. **Trigger**: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error.

Incoming links (from one component to this one): **Row**: Iterate. **Trigger**: Run if; On Subjob Ok; On Subjob Error; On component Ok; On Component Error; Synchronize; Parallelize.

For further information regarding connections, see *Talend Studio User Guide*.

Scenario: Displaying the properties of a processed file

This Java scenario describes a very simple Job that displays the properties of the specified file.

Procedure

1. Drop a **tFileProperties** component and a **tLogRow** component from the **Palette** onto the design workspace.
2. Right-click on **tFileProperties** and connect it to **tLogRow** using a **Main Row** link.
3. In the design workspace, select **tFileProperties**.
4. Click the **Component** tab to define the basic settings of **tFileProperties**.

![tFileProperties_1](image)

<table>
<thead>
<tr>
<th>Basic settings</th>
<th>Schema</th>
<th>Built-In</th>
<th>Edit schema</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced settings</td>
<td>File</td>
<td>&quot;C:\test5.txt&quot;</td>
<td>*</td>
<td>...</td>
</tr>
<tr>
<td>Dynamic settings</td>
<td>Calculate MD5 Hash</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. Set **Schema** type to **Built-In**.
6. If desired, click the **Edit schema** button to see the read-only columns.
7. In the **File** field, enter the file path or browse to the file you want to display the properties for.
8. In the design workspace, select **tLogRow** and click the **Component** tab to define its basic settings. For more information, see **tLogRow** on page 1988.
9. Press **F6** to execute the Job.

```plaintext
| #1. tLogRow_1 |
| key | value |
| abs_path | C:\test5.txt |
| basename | test5.txt |
| node_string | rv |
| size | 06 |
| ntime | 1219674736421 |
| ntime_string | Mon Aug 25 16:32:16 CEST 2008 |

Job File_Properties ended at 14:57 26/08/2008. [exit code=0]
```

**Results**

The properties of the defined file are displayed on the console.
tFileRowCount

Opens a file and reads it row by row in order to determine the number of rows inside.

**tFileRowCount Standard properties**

These properties are used to configure tFileRowCount running in the Standard Job framework.

The Standard tFileRowCount component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>File Name</th>
<th>Name or path to the file to be processed and/or the variable to be used. For further information about how to define and use a variable in a Job, see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row separator</td>
<td>String (ex: &quot;\n&quot; on Unix) to distinguish rows in the output file.</td>
</tr>
<tr>
<td>Ignore empty rows</td>
<td>Select this check box to ignore the empty rows while the component is counting the rows in the file.</td>
</tr>
<tr>
<td>Encoding</td>
<td>Select the encoding type from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for DB data handling.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | **COUNT**: the number of rows in a file. This is a Flow variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tFileRowCount is a standalone component, it must be used with a OnSubjobOk connection to tJava.</th>
</tr>
</thead>
</table>

### Connections

**Outgoing links (from this component to another):**
- **Row:** Main; Iterate.
- **Trigger:** On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error.

**Incoming links (from one component to this one):**
- **Row:** Main; Reject; Iterate.
- **Trigger:** On Subjob Ok; On Subjob Error; Run if; On component Ok; On Component Error; Synchronize; Parallelize.

For further information regarding connections, see Talend Studio User Guide.

### Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

### Scenario: Writing a file to MySQL if the number of its records matches a reference value

In this scenario, tFileRowCount counts the number of records in a .txt file, which is compared against a reference value through tJava. Once the two values match, the .txt file will be written to a MySQL table.

The .txt file has two records:

```
1;andy
2;mike
```

### Linking the components

**Procedure**

1. Drop tFileRowCount, tJava, tFlieInputDelimited, and tMysqlOutput from the Palette onto the design workspace.
2. Link tFileRowCount to tJava using an OnSubjobOk trigger.
3. Link tJava to tFlieInputDelimited using a Run if trigger.
4. Link tFlieInputDelimited to tMysqlOutput using a Row > Main connection.
Configuring the components

Procedure

1. Double-click **tFileRowCount** to open its **Basic settings** view.

2. In the **File Name** field, type in the full path of the .txt file. You can also click the [...] button to browse for this file.

   Select the **Ignore empty rows** check box.

3. Double-click **tJava** to open its **Basic settings** view.

   In the **Code** box, enter the function to print out the number of rows in the file:

   ```java
   System.out.println(globalMap.get("tFileRowCount_1_COUNT"));
   ```

4. Click the **if** trigger connection to open its **Basic settings** view.
In the **Condition** box, enter the statement to judge if the number of rows is 2:

```java
((Integer)globalMap.get("tFileRowCount_1_COUNT")) == 2
```

This if trigger means that if the row count equals 2, the rows of the .txt file will be written to MySQL.

5. Double-click **tFileInputDelimited** to open its **Basic settings** view.

In the **File name/Stream** field, type in the full path of the .txt file. You can also click the `[...]` button to browse for this file.

6. Click the **Edit schema** button open the schema editor.

7. Click the `[+]` button to add two columns, namely *id* and *name*, respectively of the *integer* and *string* type.

8. Click the **Yes** button in the pop-up box to propagate the schema setup to the following component.
9. Double-click **tMysqlOutput** open its **Basic settings** view.

10. In the **Host** and **Port** fields, enter the connection details.
    In the **Database** field, enter the database name.
    In the **Username** and **Password** fields, enter the authentication details.
    In the **Table** field, enter the table name, for instance "staff".

11. In the **Action on table** list, select **Create table if not exists**.
    In the **Action on data** list, select **Insert**.

**Executing the Job**

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to run the Job.

```
Starting job tFileRowCount at 10:35 27/08/2013

[statistics] connecting to socket on port 3610
[statistics] connected
2
[statistics] disconnected
Job tFileRowCount ended at 10:35 27/08/2013 [exit code=0]
```
As shown above, the Job has been executed successfully and the number of rows in the .txt file has been printed out.

3. Go to the MySQL GUI and open the table *staff*.

As shown above, the table has been created with the two records inserted.
tFileTouch

Creates an empty file or, if the specified file already exists, updates its date of modification and of last access while keeping the contents unchanged.

tFileTouch Standard properties

These properties are used to configure tFileTouch running in the Standard Job framework.

The Standard tFileTouch component belongs to the File family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>File Name</th>
<th>Path and name of the file to be created and/or the variable to be used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create directory if not exists</td>
<td>This check box is selected by default. It creates a directory to hold the output table if it does not exist.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component can be used as a standalone component. |
| Connections | Outgoing links (from this component to another): Row: Main. |
### tFileTouch

<table>
<thead>
<tr>
<th><strong>Trigger:</strong></th>
<th>On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error.</th>
</tr>
</thead>
</table>

Incoming links (from one component to this one):

| **Row:** | Main; Reject; Iterate. |

**Trigger:** Run if; On Subjob Ok; On Subjob Error; On component Ok; On Component Error; Synchronize; Parallelize.

For further information regarding connections, see *Talend Studio User Guide*.

## Related scenarios

No scenario is available for the Standard version of this component yet.
**tFileUnarchive**

Decompresses an archive file for further processing, in one of the following formats: *.tar.gz, *.tgz, *.tar, *.gz and *.zip.

**tFileUnarchive Standard properties**

These properties are used to configure tFileUnarchive running in the Standard Job framework.

The Standard tFileUnarchive component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive file</td>
<td>File path to the archive.</td>
</tr>
<tr>
<td>Extraction Directory</td>
<td>Folder where the unzipped file(s) will be put.</td>
</tr>
<tr>
<td>Use archive file name as root directory</td>
<td>Select this check box to create a folder named as the archive, if it does not exist, under the specified directory and extract the zipped file(s) to that folder.</td>
</tr>
<tr>
<td>Check the integrity before unzip</td>
<td>Select this check box to run an integrity check before unzipping the archive.</td>
</tr>
<tr>
<td>Extract file paths</td>
<td>Select this check box to reproduce the file path structure zipped in the archive.</td>
</tr>
<tr>
<td>Need a password</td>
<td>Select this check box and provide the correct decrypt method and password if the archive to be unzipped is password protected. Note that the encrypted archive must be one created by the tFileArchive component; otherwise you will see error messages or get nothing extracted even if no error message is displayed. Decrypt method: select the decrypt method from the list, either Java Decrypt or Zip4j Decrypt. Enter the password: enter the decryption password. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Variables</td>
<td>CURRENT_FILE: the current file name. This is a Flow variable and it returns a string.</td>
</tr>
</tbody>
</table>
CURRENT_FILEPATH: the current file path. This is a Flow variable and it returns a string.

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

#### Usage rule

This component can be used as a standalone component but it can also be used within a Job as a Start component using an **Iterate** link.

#### Connections

Outgoing links (from this component to another):

**Row**: Iterate.

**Trigger**: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error.

Incoming links (from one component to this one):

**Row**: Iterate.

**Trigger**: Run if; On Subjob Ok; On Subjob Error; On component Ok; On Component Error; Synchronize; Parallelize.

For further information regarding connections, see *Talend Studio User Guide*.

#### Limitation

**Warning:**


### Related scenario

For **tFileUnarchive** related scenario, see **tFileCompare** on page 980.
tFilterColumns

Homogenizes schemas either by ordering the columns, removing unwanted columns or adding new columns.

**tFilterColumns Standard properties**

These properties are used to configure tFilterColumns running in the Standard Job framework.

The Standard tFilterColumns component belongs to the Processing family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.  
Click **Sync columns** to retrieve the schema from the previous component in the Job. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |
Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule | This component is not startable (green background) and it requires an output component. |

Related Scenario

For more information regarding the tFilterColumns component in use, see Scenario 1: Multiple replacements and column filtering on page 3173.
tFilterRow

Filters input rows by setting one or more conditions on the selected columns.

**tFilterRow Standard properties**

These properties are used to configure tFilterRow running in the Standard Job framework.

The Standard tFilterRow component belongs to the Processing family.

The component in this framework is generally available.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

The schema of this component is built-in only. |
| **Logical operator used to combine conditions** | Select a logical operator to combine simple conditions and to combine the filter results of both modes if any advanced conditions are defined.

**And**: returns the boolean value of `true` if all conditions are true; otherwise `false`. For each two conditions combined using a logical AND, the second condition is evaluated only if the first condition is evaluated to be true.

**Or**: returns the boolean value of `true` if any condition is true; otherwise `false`. For each two conditions combined using a logical OR, the second condition is evaluated only if the first condition is evaluated to be false. |
| **Conditions** | Click the plus button to add as many simple conditions as needed. Based on the logical operator selected, the conditions are evaluated one after the other in sequential order for each row. When evaluated, each condition returns the boolean value of `true` or `false`.

**Input column**: Select the column of the schema the function is to be operated on

**Function**: Select the function on the list

**Operator**: Select the operator to bind the input column with the value

**Value**: Type in the filtered value, between quotes if needed. |
| **Use advanced mode** | Select this check box when the operations you want to perform cannot be carried out through the standard functions offered, for example, different logical operations in the same component. In the text field, type in the regular expression as required. |
If multiple advanced conditions are defined, use a logical operator between two conditions:

\& \& (logical AND): returns the boolean value of true if both conditions are true; otherwise false. The second condition is evaluated only if the first condition is evaluated to be true.

\| \| (logical OR): returns the boolean value of true if either condition is true; otherwise false. The second condition is evaluated only if the first condition is evaluated to be false.

### Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_OK: the number of rows matching the filter. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_REJECTED: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

| Usage rule | This component is not startable (green background) and it requires an output component. |

### Scenario 1: Filtering a list of names using simple conditions

The following scenario shows a Job that uses simple conditions to filter a list of records. This scenario will output two tables: the first will list all male persons with a last name shorter than nine characters
and aged between 10 and 80 years; the second will list all rejected records. An error message for each rejected record will display in the same table to explain why such a record has been rejected.

**Dropping and linking components**

**Procedure**

1. Drop `tFixedFlowInput`, `tFilterRow` and `tLogRow` from the Palette onto the design workspace.
2. Connect the `tFixedFlowInput` to the `tFilterRow`, using a Row > Main link. Then, connect the `tFilterRow` to the `tLogRow`, using a Row > Filter link.
3. Drop `tLogRow` from the Palette onto the design workspace and rename it as `reject`. Then, connect the `tFilterRow` to the `reject`, using a Row > Reject link.
4. Label the components to better identify their roles in the Job.

**Configuring the components**

**Procedure**

1. Double-click `tFixedFlowInput` to display its Basic settings view and define its properties.
2. Click the [...] button next to Edit schema to define the schema for the input data. In this example, the schema is made of the following four columns: `LastName` (type String), `Gender` (type String), `Age` (type Integer) and `City` (type String).
When done, click **OK** to validate the schema setting and close the dialog box. A new dialog box opens and asks you if you want to propagate the schema. Click **Yes**.

3. Set the row and field separators in the corresponding fields if needed. In this example, use the default settings for both, namely the row separator is a carriage return and the field separator is a semi-colon.

4. Select the **Use Inline Content(delimited file)** option in the **Mode** area and type in the input data in the **Content** field.

![Input dialog box](image)

The input data used in this example is shown below:

```
Van Buren;M;73;Chicago
Adams;M;40;Albany
Jefferson;F;66;New York
Adams;M;9;Albany
Jefferson;M;30;Chicago
Carter;F;26;Chicago
Harrison;M;40;New York
Roosevelt;F;15;Chicago
Monroe;M;8;Boston
Arthur;M;20;Albany
Pierce;M;18;New York
Quincy;F;83;Albany
McKinley;M;70;Boston
Coolidge;M;60;Chicago
Monroe;M;60;Chicago
```

5. Double-click **tFilterRow** to display its **Basic settings** view and define its properties.
6. In the **Conditions** table, add four conditions and fill in the filtering parameters.

   - From the **InputColumn** list field of the first row, select *LastName*, from the **Function** list field, select *Length*, from the **Operator** list field, select *Lower than*, and in the **Value** column, type in 9 to limit the length of last names to nine characters.

   - From the **InputColumn** list field of the second row, select *Gender*, from the **Operator** list field, select *Equals*, and in the **Value** column, type in "M" in double quotes to filter records of male persons.

   **Warning:**
   
   In the **Value** field, you must type in your values between double quotes for all types of values, except for integer values, which do not need quotes.

   - From the **InputColumn** list field of the third row, select *Age*, from the **Operator** list field, select *Greater than*, and in the **Value** column, type in 10 to set the lower limit to 10 years.

   - From the **InputColumn** list field of the four row, select *Age*, from the **Operator** list field, select *Lower than*, and in the **Value** column, type in 80 to set the upper limit to 80 years.

7. To combine the conditions, select **And** as that only those records that meet all the defined conditions are accepted.

8. In the **Basic settings** of **tLogRow** components, select **Table (print values in cells of a table)** in the **Mode** area.

**Executing the Job**

**Procedure**

Save your Job and press **F6** to execute it.
As shown above, the first table lists the records of male persons aged between 10 and 80 years, whose last names are made up of less than nine characters, and the second table lists all the records that do not match the filter conditions. Each rejected record has a corresponding error message that explains the reason of rejection.

### Scenario 2: Filtering a list of names through different logical operations

Based on the previous scenario, this scenario further filters the input data so that only those records of people from New York and Chicago are accepted. Without changing the filter settings defined in the previous scenario, advanced conditions are added in this scenario to enable both logical AND and logical OR operations in the same `tFilterRow` component.

#### Procedure

1. Double-click the `tFilterRow` component to show its Basic settings view.
2. Select the **Use advanced mode** check box, and type in the following expression in the text field:

```
input_row.City.equals("Chicago") || input_row.City.equals("New York")
```

This defines two conditions on the *City* column of the input data to filter records that contain the cities of Chicago and New York, and uses a logical OR to combine the two conditions so that records satisfying either condition will be accepted.

3. Press **Ctrl+S** to save the Job and press **F6** to execute it.

As shown above, the result list of the previous scenario has been further filtered, and only the records containing the cities of New York and Chicago are accepted.
tFirebirdClose

Closes a transaction with a Firebird database.

**tFirebirdClose Standard properties**

These properties are used to configure tFirebirdClose running in the Standard Job framework.

The Standard tFirebirdClose component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tFirebirdConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with Firebird components, especially with tFirebirdConnection and tFirebirdCommit.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-</td>
</tr>
</tbody>
</table>
Related scenarios

No scenario is available for the Standard version of this component yet.
tFirebirdCommit

Commits a global transaction instead of doing so on every row or every batch, thus providing a gain in performance.

tFirebirdCommit Standard properties

These properties are used to configure tFirebirdCommit running in the Standard Job framework.
The Standard tFirebirdCommit component belongs to the Databases family.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tFirebirdConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
<tr>
<td></td>
<td>Warning: If you want to use a Row &gt; Main connection to link tFirebirdCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStat Catcher Statistics      | Select this check box to collect log data at the component level. |

Usage

| Usage rule                   | This component is more commonly used with other tFirebird* components, especially with the tFirebirdConnection and tFirebirdRollback components. |
## Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

---

### Related scenario

For **tFirebirdCommit** related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476.
tFirebirdConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

tFirebirdConnection Standard properties

These properties are used to configure tFirebirdConnection running in the Standard Job framework. The Standard tFirebirdConnection component belongs to the Databases and the ELT families.

The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

| Host name           | Database server IP address.                                                |
| Database            | Name of the database.                                                      |

<table>
<thead>
<tr>
<th>Username and Password</th>
<th>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</th>
</tr>
</thead>
</table>

Use or register a shared DB Connection

Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the Shared DB Connection Name field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the Use dynamic job and Use an independent process to run subjob options of the tRunJob component. Using a shared connection...
together with a **tRunJob** component with either of these two options enabled will cause your Job to fail.

**Advanced settings**

**Auto Commit**

Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared.

By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

**tStatCatcher Statistics**

Select this check box to gather the job processing metadata at a Job level as well as at each component level.

**Usage**

**Usage rule**

This component is more commonly used with other **tFirebird** components, especially with the **tFirebirdCommit** and **tFirebirdRollback** components.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided.

You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

**Related scenarios**

For **tFirebirdConnection** related scenario, see **tMysqlConnection** on page 2475
tFirebirdInput

Executes a database query on a Firebird database with a strictly defined order which must correspond to the schema definition then passes on the field list to the next component via a Main row link.

### tFirebirdInput Standard properties

These properties are used to configure tFirebirdInput running in the Standard Job framework.

The Standard tFirebirdInput component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components on page 598](#).

#### Basic settings

<table>
<thead>
<tr>
<th>Basic settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td>1.</td>
<td>In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td>2.</td>
<td>In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>For an example about how to share a database connection across Job levels, see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of the DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In:</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Repository:</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Query type and Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Advanced Settings</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Trim all the String/Char columns</strong></td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</td>
</tr>
<tr>
<td><strong>Trim column</strong></td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
</tbody>
</table>
### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>QUERY: the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible SQL queries for FireBird databases.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</td>
</tr>
<tr>
<td></td>
<td>The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.</td>
</tr>
<tr>
<td></td>
<td>For examples on using dynamic parameters, see <a href="#">Scenario: Reading data from databases through context-based dynamic connections on page 2497</a> and <a href="#">Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489</a>. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Limitation

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. |
You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

### Related scenarios

For related topics, see:

See also related topic: *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489.
**tFirebirdOutput**

Executes the action defined on the table in a Firebird database and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tFirebirdOutput writes, updates, makes changes or suppresses entries in a database.

**tFirebirdOutput Standard properties**

These properties are used to configure tFirebirdOutput running in the Standard Job framework.

The Standard tFirebirdOutput component belongs to the Databases family.

The component in this framework is generally available.

[Note:]

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
</tbody>
</table>
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
</tbody>
</table>

**Action on table**

- **None**: No operation is carried out.
- **Drop and create a table**: The table is removed and created again.
- **Create a table**: The table does not exist and gets created.
- **Create a table if not exists**: The table is created if it does not exist.
- **Drop a table if exists and create**: The table is removed if it already exists and created again.
- **Clear a table**: The table content is deleted.

**Action on data**

- **Insert**: Add new entries to the table. If duplicates are found, Job stops.
- **Update**: Make changes to existing entries
- **Insert or update**: Insert a new record. If the record with the given reference already exists, an update would be made.
- **Update or insert**: Update the record with the given reference. If the record does not exist, a new record would be inserted.
- **Delete**: Remove entries corresponding to the input flow.

**Warning:**

You must specify at least one column as a primary key on which the **Update** and **Delete** operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the update and delete operations. To do that: Select the **Use field**.
**options** check box and then in the **Key in update** column, select the check boxes next to the column name on which you want to base the update operation. Do the same in the **Key in delete** column for the deletion operation.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

**Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](https://help.talend.com).

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](https://help.talend.com).

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Die on error

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

### Advanced settings

**Commit every**

Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.
### Additional Columns

<table>
<thead>
<tr>
<th>Name:</th>
<th>Type in the name of the schema column to be altered or inserted as new column</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL expression:</td>
<td>Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td>Position:</td>
<td>Select Before, Replace or After following the action to be performed on the reference column.</td>
</tr>
<tr>
<td>Reference column:</td>
<td>Type in a column of reference that the tDBOutput can use to place or replace the new or altered column.</td>
</tr>
</tbody>
</table>

### Use field options

Select this check box to customize a request, especially when there is double action on data.

### Enable debug mode

Select this check box to display each step during processing entries in a database.

### Support null in "SQL WHERE" statement

Select this check box if you want to deal with the Null values contained in a DB table.

**Note:**

Make sure the Nullable check box is selected for the corresponding columns in the schema.

### Use Batch

Select this check box to activate the batch mode for data processing.

**Note:**

This check box is available only when you have selected the Insert, Update, or Delete option in the Action on data option.

### Batch Size

Specify the number of records to be processed in each batch.

This field appears only when the Use batch mode check box is selected.

### tStat Catcher Statistics

Select this check box to collect log data at the component level.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_LINE_UPDATED: the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>
**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible. This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a Firebird database. It also allows you to create a reject flow using a Row &gt; Rejects link to filter data in error. For an example of tMySQLOutput in use, see Scenario: Retrieving data in error with a Reject link on page 2529.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dynamic settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click the [*] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Related scenarios

For related topics, see:

- **Scenario: Inserting a column and altering data using tMysqlOutput** on page 2521.
**tFirebirdRollback**

Cancels the transaction committed in the connected Firebird database.

**tFirebirdRollback Standard properties**

These properties are used to configure tFirebirdRollback running in the Standard Job framework.
The Standard tFirebirdRollback component belongs to the Databases family.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tFirebirdConnection</strong> component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

### Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

### Usage

| Usage rule | This component is more commonly used with other tFirebird* components, especially with the **tFirebirdConnection** and **tFirebirdCommit** components. |
| Dynamic settings | Click the [+ ] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable. |
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenario

For tFirebirdRollback related scenario, see Scenario: Rollback from inserting data in mother/daughter tables on page 2480.
tFirebirdRow

Executes the stated SQL query on the specified Firebird database.

Depending on the nature of the query and the database, tFirebirdRow acts on the actual DB structure or on the data (although without handling data). The SQLBuilder tool helps you write easily your SQL statements.

tFirebirdRow is the specific component for this database query. The row suffix means the component implements a flow in the job design although it doesn’t provide output.

tFirebirdRow Standard properties

These properties are used to configure tFirebirdRow running in the Standard Job framework.

The Standard tFirebirdRow component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

Use an existing connection

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

Note:

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: The schema is created and stored locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
<td></td>
</tr>
<tr>
<td>- <strong>View schema</strong>: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>- <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
<td></td>
</tr>
<tr>
<td>- <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td><strong>Query type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: Fill in manually the query statement or build it graphically using SQLBuilder</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</td>
<td></td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
</tbody>
</table>
### Die on error

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

### Advanced settings

#### Propagate QUERY's recordset

Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the **use column** list.

**Note:**

This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY's recordset should be set to the type of **Object** and this component is usually followed by tParseRecordSet.

#### Use PreparedStatement

Select this check box if you want to query the database using a PreparedStatement. In the **Set PreparedStatement Parameter** table, define the parameters represented by `?` in the SQL instruction of the **Query** field in the **Basic Settings** tab.

- **Parameter Index**: Enter the parameter position in the SQL instruction.
- **Parameter Type**: Enter the parameter type.
- **Parameter Value**: Enter the parameter value.

**Note:**

This option is very useful if you need to execute the same query several times. Performance levels are increased.

#### Commit every

Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.

#### tStat Catcher Statistics

Select this check box to collect log data at the component level.

### Global Variables

**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility benefit of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
</table>

#### Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

#### Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

### Related scenarios

For related topics, see:

- Scenario: Combining two flows for selective output on page 2558
- Procedure on page 625.
- Scenario: Removing and regenerating a MySQL table index on page 2552.
**tFixedFlowInput**

Generates a fixed flow from internal variables.

**tFixedFlowInput Standard properties**

These properties are used to configure tFixedFlowInput running in the Standard Job framework.

The Standard tFixedFlowInput component belongs to the Misc family.

The component in this framework is generally available.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository.
Click **Edit schema** to make changes to the schema. If the current schema is of the Repository type, three options are available:
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

| **Built-in**: The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide. |

| **Repository**: You have already created the schema and stored it in the Repository, hence can be reused in various projects and job designs. Related topic: see Talend Studio User Guide. |

| **Mode** | From the three options, select the mode that you want to use.
**Use Single Table**: Enter the data that you want to generate in the relevant value field.
**Use Inline Table**: Add the row(s) that you want to generate.
**Use Inline Content**: Enter the data that you want to generate, separated by the separators that you have already defined in the Row and Field Separator fields. |
### Number of rows
Enter the number of lines to be generated.

<table>
<thead>
<tr>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between inverted commas, enter the values corresponding to the columns you defined in the schema dialog box via the <strong>Edit schema</strong> button.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>tStat Catcher Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
</tr>
</thead>
</table>
| **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component can be used as a start or intermediate component and thus requires an output component.</td>
</tr>
</tbody>
</table>

### Related scenarios

For related scenarios, see:

- **Scenario 2: Buffering output data on the webapp server** on page 402.
- **Scenario: Iterating on a DB table and listing its column names** on page 2470.
- **Scenario 1: Filtering a list of names using simple conditions** on page 1174.
tFlowMeter

Counts the number of rows processed in the defined flow, so this number can be caught by the tFlowMeterCatcher component for logging purposes.

**tFlowMeter Standard properties**

These properties are used to configure tFlowMeter running in the Standard Job framework.

The Standard tFlowMeter component belongs to the Logs & Errors family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Use input connection name as label</strong></th>
<th>Select this check box to reuse the name given to the input main row flow as label in the logged data.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode</strong></td>
<td>Select the type of values for the data measured: <strong>Absolute</strong>: the actual number of rows is logged <strong>Relative</strong>: a ratio (%) of the number of rows is logged. When this option is selected, a Connections List shows to let you select a reference connection.</td>
</tr>
<tr>
<td><strong>Thresholds</strong></td>
<td>Adds a threshold to watch proportions in volumes measured. you can decide that the normal flow has to be between low and top end of a row number range, and if the flow is under this low end, there is a bottleneck.</td>
</tr>
</tbody>
</table>

**Global Variables**

| **Global Variables** | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

**Usage**

| **Usage rule** | Cannot be used as a start component as it requires an input flow to operate. |
If you have a need of log, statistics and other measurement of your data flows, see *Talend Studio User Guide*.

**Related scenario**

For related scenario, see *Scenario: Catching flow metrics from a Job* on page 1207
tFlowMeterCatcher

Operates as a log function triggered by the use of a tFlowMeter component in the Job. Based on a defined schema, the tFlowMeterCatcher catches the processing volumetric from the tFlowMeter component and passes them on to the output component.

**tFlowMeterCatcher Standard properties**

These properties are used to configure tFlowMeterCatcher running in the Standard Job framework. The Standard tFlowMeterCatcher component belongs to the Logs & Errors family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description, it defines the fields to be processed and passed on to the next component. In this particular case, the schema is read-only, as this component gathers standard log information including:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moment:</td>
<td>Processing time and date</td>
</tr>
<tr>
<td>Pid:</td>
<td>Process ID</td>
</tr>
<tr>
<td>Father_pid:</td>
<td>Process ID of the father Job if applicable. If not applicable, Pid is duplicated.</td>
</tr>
<tr>
<td>Root_pid:</td>
<td>Process ID of the root Job if applicable. If not applicable, pid of current Job is duplicated.</td>
</tr>
<tr>
<td>System_pid:</td>
<td>Process id generated by the system</td>
</tr>
<tr>
<td>Project:</td>
<td>Project name, the Job belongs to.</td>
</tr>
<tr>
<td>Job:</td>
<td>Name of the current Job</td>
</tr>
<tr>
<td>Job_repository_id:</td>
<td>ID generated by the application.</td>
</tr>
<tr>
<td>Job_version:</td>
<td>Version number of the current Job</td>
</tr>
<tr>
<td>Context:</td>
<td>Name of the current context</td>
</tr>
<tr>
<td>Origin:</td>
<td>Name of the component if any</td>
</tr>
<tr>
<td>Label:</td>
<td>Label of the row connection preceding the tFlowMeter component in the Job, and that will be analyzed for volumetrics.</td>
</tr>
<tr>
<td>Count:</td>
<td>Actual number of rows being processed</td>
</tr>
<tr>
<td>Reference:</td>
<td>Number of rows passing the reference link.</td>
</tr>
</tbody>
</table>
**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

**Usage rule**

This component is the start component of a secondary Job which triggers automatically at the end of the main Job.

**Limitation**

The use of this component cannot be separated from the use of the **tFlowMeter**. For more information, see **tFlowMeter** on page 1204.

**Scenario: Catching flow metrics from a Job**

The following basic Job aims at catching the number of rows being passed in the flow processed. The measures are taken twice, once after the input component, that is, before the filtering step and once right after the filtering step, that is, before the output component.

- Drop the following components from the **Palette** to the design workspace: **tMysqlInput**, **tFlowMeter** (x2), **tMap**, **tLogRow**, **tFlowMeterCatcher** and **tFileOutputDelimited**.
• Link components using row main connections and click on the label to give consistent name throughout the Job, such as US_States from the input component and filtered_states for the output from the tMap component, for example.

• Link the tFlowMeterCatcher to the tFileOutputDelimited component using a row main link also as data is passed.

• On the tMysqlInput Component view, configure the connection properties as Repository, if the table metadata are stored in the Repository. Or else, set the Type as Built-in and configure manually the connection and schema details if they are built-in for this Job.

```markdown
<table>
<thead>
<tr>
<th>Property Type</th>
<th>Repository</th>
<th>DB (MySQL):Us_states</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Version</td>
<td>Mysql 5</td>
<td></td>
</tr>
<tr>
<td>Host</td>
<td>&quot;localhost&quot;</td>
<td></td>
</tr>
<tr>
<td>Port</td>
<td>&quot;3306&quot;</td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>&quot;us+10&quot;</td>
<td></td>
</tr>
<tr>
<td>Username</td>
<td>&quot;root&quot;</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td>&quot;********&quot;</td>
<td></td>
</tr>
<tr>
<td>Schema</td>
<td>Built-In</td>
<td></td>
</tr>
<tr>
<td>Table Name</td>
<td>&quot;states&quot;</td>
<td></td>
</tr>
<tr>
<td>Query Type</td>
<td>Built-In</td>
<td></td>
</tr>
<tr>
<td>Query</td>
<td><code>SELECT * FROM states</code></td>
<td></td>
</tr>
</tbody>
</table>
```

• The 50 States of the USA are recorded in the table states. In order for all 50 entries of the table to get selected, the query to run onto the Mysql database is as follows:

```
select * from states.
```

• Select the relevant encoding type on the Advanced settings vertical tab.

• Then select the following component which is a tFlowMeter and set its properties.

- Select the check box Use input connection name as label, in order to reuse the label you chose in the log output file (tFileOutputDelimited).

- The mode is Absolute as there is no reference flow to meter against, also no Threshold is to be set for this example.

- Then launch the tMap editor to set the filtering properties.
• For this use case, drag and drop the ID and State columns from the Input area of the tMap towards the Output area. No variable is used in this example.

![Image showing column drag and drop]

• On the Output flow area (labelled filtered_states in this example), click the arrow & plus button to activate the expression filter field.

• Drag the State column from the Input area (row2) towards the expression filter field and type in the rest of the expression in order to filter the state labels starting with the letter M. The final expression looks like: row2.State.startsWith("M")

• Click OK to validate the setting.

• Then select the second tFlowMeter component and set its properties.

![Image of tFlowMeter component settings]

• Select the check box Use input connection name as label.

• Select Relative as Mode and in the Reference connections list, select US_States as reference to be measured against.

• Once again, no threshold is used for this use case.

• No particular setting is required in the tLogRow.

• Neither does the tFlowMeterCatcher as this component’s properties are limited to a preset schema which includes typical log information.

• So eventually set the log output component (tFileOutputDelimited).
• Select the **Append** check box in order to log all **tFlowMeter** measures.

• Then save your Job and press **F6** to execute it.

---

**Starting job FlowMeterCatcher at 17:56 29/08/2007.**

19 Maine
20 Maryland
21 Massachusetts
22 Michigan
23 Minnesota
24 Mississippi
25 Missouri
26 Montana

**Job FlowMeterCatcher ended at 17:56 29/08/2007 [exit code=0]**

---

The **Run** view shows the filtered state labels as defined in the Job.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>05:56 PM</td>
<td>tFlowMeter_1</td>
<td>US_States</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>05:56 PM</td>
<td>tFlowMeter_2</td>
<td>filtered_states</td>
<td>8</td>
<td>50</td>
</tr>
</tbody>
</table>

In the delimited csv file, the number of rows shown in column **count** varies between **tFlowMeter1** and **tFlowMeter2** as the filtering has then been carried out. The **reference** column shows also this difference.
tFlowToIterate

Reads data line by line from the input flow and stores the data entries in iterative global variables.

**tFlowToIterate Standard properties**

These properties are used to configure tFlowToIterate running in the Standard Job framework.

The Standard tFlowToIterate component belongs to the Orchestration family.

The component in this framework is generally available.

**Basic settings**

| Use the default (key, value) in global variables | When selected, the system uses the default value of the global variable in the current Job. |
| Customize | key: Type in a name for the new global variable. Press Ctrl+Space to access all available variables either global or user-defined. |
| | value: Click in the cell to access a list of the columns attached to the defined global variable. |

**Global Variables**

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.
- **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.
- **CURRENT_ITERATION**: the sequence number of the current iteration. This is a Flow variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

**Usage**

| Usage rule | You cannot use this component as a start component. tFlowToIterate requires an output component. |
| Connections | Outgoing links (from this component to another): |
Scenario: Transforming data flow to a list

The following scenario describes a Job that reads a list of files from a defined input file, iterates on each of the files and displays their content row by row on the Run console.

Setting up the Job

Procedure

1. Drop the following components from the Palette onto the design workspace: two tFileInputDelimited components, a tFlowToIterate, and a tLogRow.
2. Connect the first tFileInputDelimited to tFlowToIterate using a Row > Main link, tFlowToIterate to the second tFileInputDelimited using an Iterate link, and the second tFileInputDelimited to tLogRow using a Row > Main link.

Configuring the Components

Procedure

1. Double-click the first tFileInputDelimited to display its Basic settings view.
2. Click the [...] button next to the File Name field to select the path to the input file.

Note: The File Name field is mandatory.
The input file used in this scenario is Customers.txt. It is a text file that contains a list of names of three other simple text files: Name.txt, E-mail.txt and Address.txt. The first text file, Name.txt, is made of one column holding customers’ names. The second text file, E-mail.txt, is made of one column holding customers’ e-mail addresses. The third text file, Address.txt, is made of one column holding customers’ postal addresses.

Fill in all other fields as needed. For more information, see tFileInputDelimited Standard properties on page 1010. In this scenario, the header and the footer are not set and there is no limit for the number of processed rows.

3. Click Edit schema to describe the data structure of this input file. In this scenario, the schema is made of one column, FileName.

4. Double-click tFlowToIterate to display its Basic settings view.

Click the plus button to add new parameter lines and define your variables, and click in the key cell to enter the variable name as desired. In this scenario, one variable is defined: "Name_of_File".

Alternatively, you can select the Use the default (key, value) in global variables check box to use the default in global variables.

5. Double-click the second tFileInputDelimited to display its Basic settings view.
In the **File name** field, enter the directory of the files to be read, and then press **Ctrl+Space** to select the global variable "Name_of_File". In this scenario, the syntax is as follows:

```
"C:/scenario/flow_to_iterate/"+(String)globalMap.get("Name_of_File"))
```

Click **Edit schema** to define the schema column name. In this scenario, it is **RowContent**.

Fill in all other fields as needed. For more information, see *tFileInputDelimited Standard properties on page 1010.*

**6.** In the design workspace, select the last component, **tLogRow**, and click the **Component** tab to define its basic settings.

Define your settings as needed. For more information, see *tLogRow Standard properties on page 1988.*

**Saving and executing the Job**

**Procedure**

1. Save your Job by pressing **Ctrl+S**.
2. Execute the Job by pressing **F6** or clicking **Run** on the **Run** tab.
Results

Customers' names, customers' e-mails, and customers' postal addresses appear on the console preceded by the schema column name.
tForeach

Creates a loop on a list for an iterate link.

**tForeach Standard properties**

These properties are used to configure tForeach running in the Standard Job framework.

The Standard tForeach component belongs to the Orchestration family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Values</th>
<th>Use the [+ button to add rows to the Values table. Then click on the fields to enter the list values to be iterated upon, between double quotation marks.</th>
</tr>
</thead>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect the log data at a component level.</th>
</tr>
</thead>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CURRENT_VALUE: the value currently iterated upon. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tForeach is an input component and requires an Iterate link to connect it to another component.</th>
</tr>
</thead>
</table>
Scenario: Iterating on a list and retrieving the values

This scenario describes a two component Job in which a list is created and iterated upon in a tForeach component. The values are then retrieved in a tJava component.

Setting up the Job

Procedure
1. Drop a tForeach and a tJava component onto the design workspace.
2. Link tForeach to tJava using a Row > Iterate connection.

Results

Configuring the components

Procedure
1. Double-click tForeach to open its Basic settings view:
2. Click the [+] button to add as many rows to the Values list as required.
3. Click on the Value fields to enter the list values, between double quotation marks.
4. Double-click tJava to open its Basic settings view:
5. Enter the following Java code in the Code area:

```
System.out.println(globalMap.get("tForeach_1_CURRENT_VALUE")+"_out");
```

**Saving and executing the Job**

**Procedure**

1. Press Ctrl+S to save your Job.
2. Press F6 to execute the Job.

**Results**

The tJava run view displays the list values retrieved from tForeach, each one suffixed with _out:
tFTPClose

Closes an active FTP connection to release the occupied resources.

**tFTPClose Standard properties**

These properties are used to configure tFTPClose running in the Standard Job framework.
The Standard tFTPClose component belongs to the Internet family.
The component in this framework is generally available.

**Basic settings**

| Component list | Select the component that opens the connection you need to close from the list. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

**Usage**

| Usage rule | This component is more commonly used with other FTP components, especially with the tFTPConnection component. |

**Related scenarios**

- Listing and getting files/folders on an FTP directory on page 1232
- Putting files onto an FTP server on page 1248
- Renaming a file located on an FTP server on page 1255
tFTPConnection

Opens an FTP connection to transfer files in a single transaction.

**tFTPConnection Standard properties**

These properties are used to configure tFTPConnection running in the Standard Job framework.

The Standard tFTPConnection component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>The IP address or hostname of the FTP server.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>The listening port number of the FTP server.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>The user authentication data to access the FTP server.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>SFTP Support</strong></td>
<td>Select this check box to connect to the FTP server via an SFTP connection.</td>
</tr>
<tr>
<td><strong>Authentication method</strong></td>
<td>Select the SFTP authentication method, either Public key or Password.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Public key</strong>: Enter the path to the private key and the passphrase for the key in the Private key and Key Passphrase fields correspondingly.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Password</strong>: Enter the password required.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the SFTP Support check box is selected.</td>
</tr>
<tr>
<td><strong>Filename encoding</strong></td>
<td>Select this check box to set the encoding used to convert file names from Strings to bytes. It should be the same encoding used on the SFTP server. If the SFTP server’s version is greater than 3, the encoding should be UTF-8, or else an error occurs.</td>
</tr>
</tbody>
</table>
This property is available only when the **SFTP Support** check box is selected.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tFTPConnection</strong></td>
<td></td>
</tr>
<tr>
<td><strong>FTPS Support</strong></td>
<td>Select this check box to connect to the FTP server via an FTPS connection.</td>
</tr>
<tr>
<td><strong>Keystore File</strong></td>
<td>The path to your keystore file, a password protected file containing several keys and certificates. This property is available only when the <strong>FTPS Support</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Keystore Password</strong></td>
<td>The password for your keystore file. This property is available only when the <strong>FTPS Support</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Security Mode</strong></td>
<td>Select the security mode from the list, either <em>Implicit</em> or <em>Explicit</em>. This property is available only when the <strong>FTPS Support</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Connection mode</strong></td>
<td>Select the connection mode from the list, either <em>Passive</em> or <em>Active</em>.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Specify the encoding type by selecting an encoding type from the list or selecting <strong>CUSTOM</strong> and enter the encoding type manually.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use Socks Proxy</strong></td>
<td>Select this check box if you are using a proxy, and in the <strong>Proxy host</strong>, <strong>Proxy port</strong>, <strong>Proxy user</strong> and <strong>Proxy password</strong> fields displayed, specify the proxy server settings information.</td>
</tr>
<tr>
<td><strong>Data Channel Protection Level</strong></td>
<td>The data channel protection level with which data is transferred between the client and the server. For more information, see <a href="https://tools.ietf.org/html/rfc2228">RFC 2228: FTP Security Extensions</a>.</td>
</tr>
<tr>
<td><strong>Protection Buffer Size</strong></td>
<td>The maximum size (in bytes) of the encoded data blocks to be transferred between the client and the server. For more information, see <a href="https://tools.ietf.org/html/rfc2228">RFC 2228: FTP Security Extensions</a>.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>
Global Variables

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

Usage

| Usage rule | This component is typically used as a single-component sub-job. It is used along with other FTP components. |

Related scenarios

- Listing and getting files/folders on an FTP directory on page 1232
- Putting files onto an FTP server on page 1248
- Renaming a file located on an FTP server on page 1255
tFTPDelete

Deletes files or folders in a specified directory on an FTP server.

**tFTPDelete Standard properties**

These properties are used to configure tFTPDelete running in the Standard Job framework.

The Standard tFTPDelete component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>The IP address or hostname of the FTP server.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>The listening port number of the FTP server.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>The user authentication data to access the FTP server. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Remote directory</strong></td>
<td>The directory where the files/folders to be deleted are located.</td>
</tr>
<tr>
<td><strong>Move to the current directory</strong></td>
<td>Select this check box to change the directory to the one specified in the Remote directory field. The next FTP component in the Job will take this directory as the root of the remote directory when using the same connection. This property is available only when the Use an existing connection check box is selected.</td>
</tr>
<tr>
<td><strong>SFTP Support</strong></td>
<td>Select this check box to connect to the FTP server via an SFTP connection.</td>
</tr>
</tbody>
</table>
| **Authentication method** | Select the SFTP authentication method, either Public key or Password.  
| • Public key: Enter the path to the private key and the passphrase for the key in the Private key and Key Passphrase fields correspondingly.  
| • Password: Enter the password required.  
| This property is available only when the SFTP Support check box is selected. |
| **Filename encoding** | Select this check box to set the encoding used to convert file names from Strings to bytes. It should be the same encoding used on the SFTP server. If the SFTP server’s version is greater than 3, the encoding should be UTF-8, or else an error occurs.  
| This property is available only when the SFTP Support check box is selected. |
| **FTPS Support** | Select this check box to connect to the FTP server via an FTPS connection. |
| **Keystore File** | The path to your keystore file, a password protected file containing several keys and certificates.  
| This property is available only when the FTPS Support check box is selected. |
| **Keystore Password** | The password for your keystore file.  
| This property is available only when the FTPS Support check box is selected. |
| **Security Mode** | Select the security mode from the list, either Implicit or Explicit.  
| This property is available only when the FTPS Support check box is selected. |
| **Use Perl5 Regex Expression as Filemask** | Select this check box to use Perl5 regular expressions in the Files field as file filters. This is useful when the name of the file to be processed contains special characters such as parentheses.  
<p>| For more information about Perl5 regular expression syntax, see Perl5 Regular Expression Syntax. |
| <strong>Files</strong> | The names of the files/folders or the paths to the files/folders to be deleted. You can specify multiple files/folders in a line by using wildcards or a regular expression. |
| <strong>Target Type</strong> | Select the type of the target to be deleted, either File or Directory. |
| <strong>Connection mode</strong> | Select the connection mode from the list, either Passive or Active. |</p>
<table>
<thead>
<tr>
<th>Encoding</th>
<th>Specify the encoding type by selecting an encoding type from the list or selecting CUSTOM and enter the encoding type manually.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die on error</td>
<td>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any error and continue the Job execution process.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Use Socks Proxy</th>
<th>Select this check box if you are using a proxy, and in the Proxy host, Proxy port, Proxy user and Proxy password fields displayed, specify the proxy server settings information.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignore Failure At Quit (FTP)</td>
<td>Select this check box to ignore library closing errors or FTP closing errors.</td>
</tr>
<tr>
<td>Data Channel Protection Level</td>
<td>The data channel protection level with which data is transferred between the client and the server. For more information, see RFC 2228: FTP Security Extensions. This property is available only when the FTP Support check box is selected.</td>
</tr>
<tr>
<td>Protection Buffer Size</td>
<td>The maximum size (in bytes) of the encoded data blocks to be transferred between the client and the server. For more information, see RFC 2228: FTP Security Extensions. This property is available only when the FTPS Support check box is selected.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_FILE</td>
<td>The number of the files processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>CURRENT_STATUS</td>
<td>The execution result of the component. This is a Flow variable and it returns a string.</td>
</tr>
</tbody>
</table>

**Usage**

| Usage rule                | This component is typically used as a single-component sub-job but can also be used as an output or end object. |

1225
Related scenario

No scenario is available for this component yet.
**tFTPFileExist**

Checks if a file or a directory exists on an FTP server.

**tFTPFileExist Standard properties**

These properties are used to configure tFTPFileExist running in the Standard Job framework.

The Standard tFTPFileExist component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Host**

The IP address or hostname of the FTP server.

**Port**

The listening port number of the FTP server.

**Username and Password**

The user authentication data to access the FTP server. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

**Remote directory**

The remote directory under which the file or the directory will be checked.

**Target Type**

Select the type of the target to be checked, either **File** or **Directory**.

**File Name**

The name of the file or the path to the file to be checked.

This property is available only when **File** is selected from the **Target Type** list.

**Directory Name**

The name of the directory or the path to the directory to be checked.
This property is available only when Directory is selected from the Target Type list.

### SFTP Support
Select this check box to connect to the FTP server via an SFTP connection.

#### Authentication method
Select the SFTP authentication method, either Public key or Password.
- **Public key**: Enter the path to the private key and the passphrase for the key in the Private key and Key Passphrase fields correspondingly.
- **Password**: Enter the password required.

This property is available only when the SFTP Support check box is selected.

#### Filename encoding
Select this check box to set the encoding used to convert file names from Strings to bytes. It should be the same encoding used on the SFTP server. If the SFTP server’s version is greater than 3, the encoding should be UTF-8, or else an error occurs.

This property is available only when the SFTP Support check box is selected.

### FTPS Support
Select this check box to connect to the FTP server via an FTPS connection.

#### Keystore File
The path to your keystore file, a password protected file containing several keys and certificates.

This property is available only when the FTPS Support check box is selected.

#### Keystore Password
The password for your keystore file.

This property is available only when the FTPS Support check box is selected.

#### Security Mode
Select the security mode from the list, either Implicit or Explicit.

This property is available only when the FTPS Support check box is selected.

#### Connection mode
Select the connection mode from the list, either Passive or Active.

#### Encoding
Specify the encoding type by selecting an encoding type from the list or selecting CUSTOM and enter the encoding type manually.

**Advanced settings**

#### Use Socks Proxy
Select this check box if you are using a proxy, and in the Proxy host, Proxy port, Proxy user and Proxy password fields displayed, specify the proxy server settings information.
<table>
<thead>
<tr>
<th><strong>Ignore Failure At Quit (FTP)</strong></th>
<th>Select this check box to ignore library closing errors or FTP closing errors.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Channel Protection Level</strong></td>
<td>The data channel protection level with which data is transferred between the client and the server. For more information, see <a href="https://tools.ietf.org/html/rfc2228">RFC 2228: FTP Security Extensions</a>. This property is available only when the <strong>FTPS Support</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Protection Buffer Size</strong></td>
<td>The maximum size (in bytes) of the encoded data blocks to be transferred between the client and the server. For more information, see <a href="https://tools.ietf.org/html/rfc2228">RFC 2228: FTP Security Extensions</a>. This property is available only when the <strong>FTPS Support</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

| **ERROR_MESSAGE** | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |
| **EXISTS** | The result of whether a specified file/directory exists. This is a Flow variable and it returns a boolean. |
| **FILENAME** | The name of the file/directory processed. This is an After variable and it returns a string. |

**Usage**

| **Usage rule** | This component is typically used as a single-component sub-job but can also be used with other components. |

**Related scenario**

No scenario is available for this component yet.
tFTPFileList

Lists all files and folders directly under a specified directory based on a filemask pattern.

**tFTPFileList Standard properties**

These properties are used to configure tFTPFileList running in the Standard Job framework.

The Standard tFTPFileList component belongs to the Internet family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>The IP address or hostname of the FTP server.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>The listening port number of the FTP server.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>The user authentication data to access the FTP server. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Remote directory</strong></td>
<td>The remote directory where the files and folders to be listed are located.</td>
</tr>
<tr>
<td><strong>Move to the current directory</strong></td>
<td>Select this check box to change the directory to the one specified in the Remote directory field. The next FTP component in the Job will take this directory as the root of the remote directory when using the same connection.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the Use an existing connection check box is selected.</td>
</tr>
<tr>
<td><strong>File detail</strong></td>
<td>Select this check box to list the details of each file/folder. The informative details include the file/folder permissions, the name of the author, the name of the</td>
</tr>
<tr>
<td>File</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Files</strong></td>
<td>The names of the files/folders to be listed. You can specify multiple files/folders in a line by using wildcards or a regular expression.</td>
</tr>
<tr>
<td><strong>SFTP Support</strong></td>
<td>Select this check box to connect to the FTP server via an SFTP connection.</td>
</tr>
</tbody>
</table>
| **Authentication method** | Select the SFTP authentication method, either **Public key** or **Password**.  
  - **Public key**: Enter the path to the private key and the passphrase for the key in the **Private key** and **Key Passphrase** fields correspondingly.  
  - **Password**: Enter the password required.  
This property is available only when the **SFTP Support** check box is selected. |
| **Filename encoding** | Select this check box to set the encoding used to convert file names from Strings to bytes. It should be the same encoding used on the SFTP server. If the SFTP server’s version is greater than 3, the encoding should be **UTF-8**, or else an error occurs.  
This property is available only when the **SFTP Support** check box is selected. |
| **FTPS Support** | Select this check box to connect to the FTP server via an FTPS connection. |
| **Keystore File** | The path to your keystore file, a password protected file containing several keys and certificates.  
This property is available only when the **FTPS Support** check box is selected. |
| **Keystore Password** | The password for your keystore file.  
This property is available only when the **FTPS Support** check box is selected. |
| **Security Mode** | Select the security mode from the list, either **Implicit** or **Explicit**.  
This property is available only when the **FTPS Support** check box is selected. |
| **Connection mode** | Select the connection mode from the list, either **Passive** or **Active**. |
| **Encoding** | Specify the encoding type by selecting an encoding type from the list or selecting **CUSTOM** and enter the encoding type manually. |
Advanced settings

<table>
<thead>
<tr>
<th>Use Socks Proxy</th>
<th>Select this check box if you are using a proxy, and in the Proxy host, Proxy port, Proxy user and Proxy password fields displayed, specify the proxy server settings information.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignore Failure At Quit (FTP)</td>
<td>Select this check box to ignore library closing errors or FTP closing errors.</td>
</tr>
<tr>
<td>Data Channel Protection Level</td>
<td>The data channel protection level with which data is transferred between the client and the server. For more information, see RFC 2228: FTP Security Extensions. This property is available only when the FTPS Support check box is selected.</td>
</tr>
<tr>
<td>Protection Buffer Size</td>
<td>The maximum size (in bytes) of the encoded data blocks to be transferred between the client and the server. For more information, see RFC 2228: FTP Security Extensions. This property is available only when the FTPS Support check box is selected.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

Global Variables

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENT_FILE</td>
<td>The current file name. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>CURRENT_FILEPATH</td>
<td>The current file path. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>NB_FILE</td>
<td>The number of the files processed. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule | This component is typically used as a single-component sub-job but can also be used with other components.                                                                                       |

Listing and getting files/folders on an FTP directory

Here is an example of using Talend FTP components to iterate and list all files and folders on an FTP server directory, and then get only text files on that directory to a local directory.
Creating a Job for listing and getting files/folders on an FTP directory

Create a Job to connect to an FTP server, iterate and list all files and folders on an FTP root directory, then get only text files on the FTP root directory to a local directory, finally close the connection to the server.

Before you begin

Prerequisites: To replicate this scenario, an FTP server must be started and a couple of files/folders must be put onto the root directory of the FTP server.
**Procedure**

1. Create a new Job and add a **tFTPConnection** component, a **tFTPFileList** component, a **tIterateToFlow** component, a **tLogRow** component, a **tFTPGet** component, and a **tFTPClose** component by typing their names in the design workspace or dropping them from the **Palette**.

2. Link the **tFTPFileList** component to the **tIterateToFlow** component using a **Row > Iterate** connection.

3. Link the **tIterateToFlow** component to the **tLogRow** component using a **Row > Main** connection.

4. Link the **tFTPConnection** component to the **tFTPFileList** component using a **Trigger > OnSubjobOk** connection.

5. Do the same to link the **tFTPFileList** component to the **tFTPGet** component, and the **tFTPGet** component to the **tFTPClose** component.

### Opening a connection to the FTP server

Configure the **tFTPConnection** component to open a connection to the FTP server.

**Procedure**

1. Double-click the **tFTPConnection** component to open its **Basic settings** view.

2. In the **Host** and **Port** fields, enter the FTP server IP address and the listening port number respectively.

3. In the **Username** and **Password** fields, enter the authentication details.

### Listing all files/folders on the FTP root directory

Configure the **tFTPFileList** component, the **tIterateToFlow** component, and the **tLogRow** component to iterate all files and folders on the FTP root directory and display the names and paths of these files and folders on the console of **Talend Studio**.

**Procedure**

1. Double-click the **tFTPFileList** component to open its **Basic settings** view.

2. Specify the connection details required to access the FTP server. In this example, select the **Use an existing connection** check box and from the **Component list** drop-down list displayed, select the connection component to reuse its connection details you have already defined.
3. In the **Remote directory** field, specify the FTP server directory on which the files and folders will be iterated. In this example, it is /, which means the root directory of the FTP server.

4. Clear the **Move to the current directory** check box.

5. Double-click the **tIterateToFlow** component to open its **Basic settings** view.

6. Click the **Edit schema** button to open the schema dialog box.

7. Click the **+** button to add two String type columns *filename* and *filepath* that will hold the names and paths of the files to be iterated respectively. When done, click **OK** to close the dialog box.

8. In the **Mapping** table, set the values for the *filename* and *filepath* columns. In this example, the global variable `((String)globalMap.get("tFTPFileList_1_CURRENT_FILE"))` for *filename* and the global variable `((String)globalMap.get("tFTPFileList_1_CURRENT_FILEPATH"))` for *filepath*.

   Note that you can fill the values by pressing **Ctrl + Space** to access the global variables list and then selecting `tFTPFileList_1_CURRENT_FILE` and `tFTPFileList_1_CURRENT_FILEPATH` from the list.

9. Double-click the **tLogRow** component to open its **Basic settings** view, and then select **Table (print values in cells of a table)** in the **Mode** area for better readability of the result.
Getting files on the FTP server directory to a local directory

Configure the tFTPGet component to get only the text files on the FTP root directory to a local directory.

Procedure

1. Double-click the tFTPGet component to open its Basic settings view.

2. Specify the connection details required to access the FTP server. In this example, select the Use an existing connection check box and from the Component list drop-down list displayed, select the connection component to reuse its connection details you have already defined.

3. In the Local directory field, specify the local directory to which the files and folders will be downloaded. In this example, it is D:/FtpDownloads.

4. In the Remote directory field, specify the FTP server directory under which the files and folders will be downloaded. In this example, it is /, which means the root directory of the FTP server.

5. In the Files table, click the [+] button to add a line and in the Filemask column field, enter *.txt between double quotation marks to get only the text files on the FTP directory to the local directory.

Closing the connection to the FTP server

Configure the tFTPClose component to close the connection to the FTP server.

Procedure

1. Double-click the tFTPClose component to open its Basic settings view.
2. From the Component list drop-down list, select the tFTPConnection component that opens the connection you need to close. In this example, only one tFTPConnection component is used and it is selected by default.

**Executing the Job to list and get files/folders on the FTP directory**

After setting up the Job and configuring the components used in the Job for listing and getting files/folders on the FTP directory, you can then execute the Job and verify the Job execution result.

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to execute the Job.

   ```
   [statistics] connecting to socket on port 4009
   [statistics] connected
   tLogBox_1
<table>
<thead>
<tr>
<th>filename</th>
<th>filepath</th>
</tr>
</thead>
<tbody>
<tr>
<td>employee.txt</td>
<td>//employee.txt</td>
</tr>
<tr>
<td>product.txt</td>
<td>//product.txt</td>
</tr>
<tr>
<td>TempFolder</td>
<td>//TempFolder</td>
</tr>
<tr>
<td>tFTPConnection_icon32.png</td>
<td>//tFTPConnection_icon32.png</td>
</tr>
</tbody>
</table>
   [statistics] disconnected
   ```

As shown above, the names and paths of the files and folders on the FTP server root directory are displayed on the console, and only the text files are downloaded to the specified local directory.
tFTPFileProperties

Retrieves the properties of a specified file on an FTP server.

**tFTPFileProperties Standard properties**

These properties are used to configure tFTPFileProperties running in the Standard Job framework.
The Standard tFTPFileProperties component belongs to the Internet family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A schema is a row description, and it defines the fields to be processed and passed on to the next component.</td>
<td></td>
</tr>
<tr>
<td>The schema of this component is read-only. It describes the main properties of the specified file. You can click the [...] button next to Edit schema to view the predefined schema which contains the following fields:</td>
<td></td>
</tr>
<tr>
<td>• <strong>abs_path</strong>: the absolute path of the file.</td>
<td></td>
</tr>
<tr>
<td>• <strong>dirname</strong>: the directory of the file.</td>
<td></td>
</tr>
<tr>
<td>• <strong>basename</strong>: the name of the file.</td>
<td></td>
</tr>
<tr>
<td>• <strong>size</strong>: the file size in bytes.</td>
<td></td>
</tr>
<tr>
<td>• <strong>mtime</strong>: the timestamp indicating when the file was last modified, in milliseconds that have elapsed since the Unix epoch (00:00:00 UTC, Jan 1, 1970).</td>
<td></td>
</tr>
<tr>
<td>• <strong>mtime_string</strong>: the date and time the file was last modified.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
<td></td>
</tr>
</tbody>
</table>

| Host | The IP address or hostname of the FTP server. |
| Port | The listening port number of the FTP server. |
| Username and Password | The user authentication data to access the FTP server. |
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

<table>
<thead>
<tr>
<th>Remote directory</th>
<th>The path to the directory where the file is available.</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>The name of the file or the path to the file whose properties will be retrieved.</td>
</tr>
<tr>
<td>Transfer mode</td>
<td>Select the transfer mode from the list, either ascii or binary.</td>
</tr>
<tr>
<td>SFTP Support</td>
<td>Select this check box to connect to the FTP server via an SFTP connection.</td>
</tr>
</tbody>
</table>
| Authentication method | Select the SFTP authentication method, either Public key or Password.  
                         - Public key: Enter the path to the private key and the passphrase for the key in the Private key and Key Passphrase fields correspondingly.  
                         - Password: Enter the password required.  
                         This property is available only when the SFTP Support check box is selected. |
| Filename encoding | Select this check box to set the encoding used to convert file names from Strings to bytes. It should be the same encoding used on the SFTP server. If the SFTP server’s version is greater than 3, the encoding should be UTF-8, or else an error occurs.  
                         This property is available only when the SFTP Support check box is selected. |
| FTPS Support     | Select this check box to connect to the FTP server via an FTPS connection. |
| Keystore File    | The path to your keystore file, a password protected file containing several keys and certificates.  
                         This property is available only when the FTPS Support check box is selected. |
| Keystore Password | The password for your keystore file.  
                         This property is available only when the FTPS Support check box is selected. |
| Security Mode    | Select the security mode from the list, either Implicit or Explicit.  
                         This property is available only when the FTPS Support check box is selected. |
| Connection mode  | Select the connection mode from the list, either Passive or Active. |
### Encoding
Specify the encoding type by selecting an encoding type from the list or selecting **CUSTOM** and enter the encoding type manually.

### Calculate MD5 Hash
Select this check box to check the file's MD5.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use Socks Proxy</strong></td>
<td>Select this check box if you are using a proxy, and in the <strong>Proxy host</strong>, <strong>Proxy port</strong>, <strong>Proxy user</strong> and <strong>Proxy password</strong> fields displayed, specify the proxy server settings information.</td>
</tr>
<tr>
<td><strong>Ignore Failure At Quit (FTP)</strong></td>
<td>Select this check box to ignore library closing errors or FTP closing errors.</td>
</tr>
<tr>
<td><strong>Data Channel Protection Level</strong></td>
<td>The data channel protection level with which data is transferred between the client and the server. For more information, see RFC 2228: FTP Security Extensions. This property is available only when the <strong>FTPS Support</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Protection Buffer Size</strong></td>
<td>The maximum size (in bytes) of the encoded data blocks to be transferred between the client and the server. For more information, see RFC 2228: FTP Security Extensions. This property is available only when the <strong>FTPS Support</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

### Usage

| Usage rule          | This component can be used as standalone component. |

### Related scenario

**Scenario:** Displaying the properties of a processed file on page 1159
tFTPGet

Downloads files to a local directory from an FTP directory.

**tFTPGet Standard properties**

These properties are used to configure tFTPGet running in the Standard Job framework.

The Standard tFTPGet component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Property Type | Select the way the connection details will be set.  
- **Built-In**: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.  
- **Repository**: The connection details stored centrally in Repository > Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in. |

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. |

| Host | The IP address or hostname of the FTP server. |
| Port | The listening port number of the FTP server. |
| Username and Password | The user authentication data to access the FTP server. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Local directory | The local directory in which downloaded files will be saved. |
| Remote directory | The FTP directory from which files will be downloaded. |
| Move to the current directory | Select this check box to change the directory to the one specified in the Remote directory field. The next FTP component in the Job will take this directory as the root of the remote directory when using the same connection.  
This property is available only when the Use an existing connection check box is selected. |
<table>
<thead>
<tr>
<th>Transfer mode</th>
<th>Select the transfer mode from the list, either <em>ascii</em> or <em>binary</em>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overwrite file</td>
<td>Select the action to be performed when the file already exists.</td>
</tr>
<tr>
<td></td>
<td>• <strong>never</strong>: Never overwrite the file.</td>
</tr>
<tr>
<td></td>
<td>• <strong>always</strong>: Always overwrite the file.</td>
</tr>
<tr>
<td></td>
<td>• <strong>size different</strong>: Overwrite the file when the file size is different.</td>
</tr>
<tr>
<td></td>
<td>• <strong>overwrite</strong>: Overwrite the existing file.</td>
</tr>
<tr>
<td></td>
<td>• <strong>resume</strong>: Resume downloading the file from the point of interruption.</td>
</tr>
<tr>
<td></td>
<td>• <strong>append</strong>: Add data to the end of the file without overwriting data.</td>
</tr>
<tr>
<td></td>
<td><strong>overwrite</strong>, <strong>resume</strong>, and <strong>append</strong> are available when the <strong>SFTP Support</strong> check box is selected.</td>
</tr>
<tr>
<td>Append</td>
<td>Select this check box to append data at the end of the file in order to avoid overwriting data.</td>
</tr>
<tr>
<td>SFTP Support</td>
<td>Select this check box to connect to the FTP server via an SFTP connection.</td>
</tr>
<tr>
<td>Authentication method</td>
<td>Select the SFTP authentication method, either <strong>Public key</strong> or <strong>Password</strong>.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Public key</strong>: Enter the path to the private key and the passphrase for the key in the <strong>Private key</strong> and <strong>Key Passphrase</strong> fields correspondingly.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Password</strong>: Enter the password required.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the <strong>SFTP Support</strong> check box is selected.</td>
</tr>
<tr>
<td>Filename encoding</td>
<td>Select this check box to set the encoding used to convert file names from Strings to bytes. It should be the same encoding used on the SFTP server. If the SFTP server’s version is greater than 3, the encoding should be <strong>UTF-8</strong>, or else an error occurs.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the <strong>SFTP Support</strong> check box is selected.</td>
</tr>
<tr>
<td>FTPS Support</td>
<td>Select this check box to connect to the FTP server via an FTPS connection.</td>
</tr>
<tr>
<td>Keystore File</td>
<td>The path to your keystore file, a password protected file containing several keys and certificates.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the <strong>FTPS Support</strong> check box is selected.</td>
</tr>
<tr>
<td>Keystore Password</td>
<td>The password for your keystore file.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the <strong>FTPS Support</strong> check box is selected.</td>
</tr>
</tbody>
</table>
| Security Mode       | Select the security mode from the list, either **Implicit** or **Explicit**.  
|                    | This property is available only when the **FTPS Support** check box is selected. |
| Use Perl5 Regex Expression as Filemask | Select this check box to use Perl5 regular expressions in the **Files** field as file filters. This is useful when the name of the file to be processed contains special characters such as parentheses.  
|                    | For more information about Perl5 regular expression syntax, see [Perl5 Regular Expression Syntax](#). |
| Files              | The names of the files or the paths to the files to be downloaded. You can specify multiple files in a line by using wildcards or a regular expression. |
| Connection mode    | Select the connection mode from the list, either **Passive** or **Active**. |
| Encoding           | Specify the encoding type by selecting an encoding type from the list or selecting **CUSTOM** and enter the encoding type manually. |
| Die on error       | Select the check box to stop the execution of the Job when an error occurs.  
|                    | Clear the check box to skip any error and continue the Job execution process. |

**Advanced settings**

| Use Socks Proxy    | Select this check box if you are using a proxy, and in the **Proxy host**, **Proxy port**, **Proxy user** and **Proxy password** fields displayed, specify the proxy server settings information. |
| Ignore Failure At Quit (FTP) | Select this check box to ignore library closing errors or FTP closing errors. |
| Print message      | Select this check box to display the list of files downloaded on the console. |
| Data Channel Protection Level | The data channel protection level with which data is transferred between the client and the server. For more information, see [RFC 2228: FTP Security Extensions](#).  
|                    | This property is available only when the **FTPS Support** check box is selected. |
| Protection Buffer Size | The maximum size (in bytes) of the encoded data blocks to be transferred between the client and the server. For more information, see [RFC 2228: FTP Security Extensions](#).  
|                    | This property is available only when the **FTPS Support** check box is selected. |
t FTPGet

### tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>NB_FILE</td>
<td>The number of the files processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>CURRENT_STATUS</td>
<td>The execution result of the component. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>TRANSFER_MESSAGES</td>
<td>The file transferred information. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This component is typically used as a single-component sub-job but can also be used as output or end object.</td>
</tr>
</tbody>
</table>

### Related scenario

Listing and getting files/folders on an FTP directory on page 1232
tFTPPut

Uploads files from a local directory to an FTP directory.

**tFTPPut Standard properties**

These properties are used to configure tFTPPut running in the Standard Job framework.

The Standard tFTPPut component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>The IP address or hostname of the FTP server.</td>
</tr>
<tr>
<td>Port</td>
<td>The listening port number of the FTP server.</td>
</tr>
</tbody>
</table>

### Username and Password

- The user authentication data to access the FTP server.
- To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

<table>
<thead>
<tr>
<th>Local directory</th>
<th>The local directory from which the files will be uploaded to the FTP server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote directory</td>
<td>The FTP directory where the uploaded files will be placed.</td>
</tr>
</tbody>
</table>

**Move to the current directory**

- Select this check box to change the directory to the one specified in the **Remote directory** field. The next FTP component in the Job will take this directory as the root of the remote directory when using the same connection.
- This property is available only when the **Use an existing connection** check box is selected.
<table>
<thead>
<tr>
<th><strong>Transfer mode</strong></th>
<th>Select the transfer mode from the list, either ascii or binary.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overwrite file</strong></td>
<td>Select the action to be performed when the file already exists.</td>
</tr>
<tr>
<td></td>
<td>• never: Never overwrite the file.</td>
</tr>
<tr>
<td></td>
<td>• always: Always overwrite the file.</td>
</tr>
<tr>
<td></td>
<td>• size different: Overwrite the file when the file size is different.</td>
</tr>
<tr>
<td></td>
<td>• overwrite: Overwrite the existing file.</td>
</tr>
<tr>
<td></td>
<td>• resume: Resume downloading the file from the point of interruption.</td>
</tr>
<tr>
<td></td>
<td>• append: Add data to the end of the file without overwriting data.</td>
</tr>
<tr>
<td></td>
<td>overwrite, resume, and append are available when the SFTP Support check box is selected.</td>
</tr>
<tr>
<td><strong>Append</strong></td>
<td>Select this check box to append data at the end of the file in order to avoid overwriting data.</td>
</tr>
<tr>
<td><strong>SFTP Support</strong></td>
<td>Select this check box to connect to the FTP server via an SFTP connection.</td>
</tr>
<tr>
<td><strong>Authentication method</strong></td>
<td>Select the SFTP authentication method, either Public key or Password.</td>
</tr>
<tr>
<td></td>
<td>• Public key: Enter the path to the private key and the passphrase for the key in the Private key and Key Passphrase fields correspondingly.</td>
</tr>
<tr>
<td></td>
<td>• Password: Enter the password required.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the SFTP Support check box is selected.</td>
</tr>
<tr>
<td><strong>Filename encoding</strong></td>
<td>Select this check box to set the encoding used to convert file names from Strings to bytes. It should be the same encoding used on the SFTP server. If the SFTP server’s version is greater than 3, the encoding should be UTF-8, or else an error occurs.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the SFTP Support check box is selected.</td>
</tr>
<tr>
<td><strong>FTPS Support</strong></td>
<td>Select this check box to connect to the FTP server via an FTPS connection.</td>
</tr>
<tr>
<td><strong>Keystore File</strong></td>
<td>The path to your keystore file, a password protected file containing several keys and certificates.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the FTPS Support check box is selected.</td>
</tr>
<tr>
<td><strong>Keystore Password</strong></td>
<td>The password for your keystore file.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the FTPS Support check box is selected.</td>
</tr>
<tr>
<td>Security Mode</td>
<td>Select the security mode from the list, either Implicit or Explicit. This property is available only when the FTPS Support check box is selected.</td>
</tr>
<tr>
<td>Use Perl5 Regex Expression as Filemask</td>
<td>Select this check box to use Perl5 regular expressions in the Files field as file filters. This is useful when the name of the file to be processed contains special characters such as parentheses. For more information about Perl5 regular expression syntax, see Perl5 Regular Expression Syntax.</td>
</tr>
<tr>
<td>Use Perl5 Regex Expression as Filemask</td>
<td>Select this check box to use Perl5 regular expressions in the Files field as file filters. This is useful when the name of the file to be processed contains special characters such as parentheses. For more information about Perl5 regular expression syntax, see Perl5 Regular Expression Syntax.</td>
</tr>
</tbody>
</table>
| Files | Specify the files to be uploaded.  
- **Filemask**: the file names or the path to the files to be uploaded.  
- **New name**: the name to give the file after the transfer. |
| Connection mode | Select the connection mode from the list, either Passive or Active. |
| Encoding | Specify the encoding type by selecting an encoding type from the list or selecting CUSTOM and enter the encoding type manually. |
| Die on error | Select the check box to stop the execution of the Job when an error occurs.  
Clear the check box to skip any error and continue the Job execution process. |
| Advanced settings |  
| Use Socks Proxy | Select this check box if you are using a proxy, and in the Proxy host, Proxy port, Proxy user and Proxy password fields displayed, specify the proxy server settings information. |
| Ignore Failure At Quit (FTP) | Select this check box to ignore library closing errors or FTP closing errors. |
| Data Channel Protection Level | The data channel protection level with which data is transferred between the client and the server. For more information, see RFC 2228: FTP Security Extensions. This property is available only when the FTPS Support check box is selected. |
**Protection Buffer Size**

The maximum size (in bytes) of the encoded data blocks to be transferred between the client and the server. For more information, see RFC 2228: FTP Security Extensions.

This property is available only when the FTPS Support check box is selected.

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td><strong>NB_FILE</strong></td>
<td>The number of the files processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>CURRENT_STATUS</strong></td>
<td>The execution result of the component. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td><strong>CURRENT_FILE_EXISTS</strong></td>
<td>The result of whether the current file exists. This is a Flow variable and it returns a boolean.</td>
</tr>
<tr>
<td><strong>TRANSFER_MESSAGES</strong></td>
<td>The file transferred information. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

### Usage

| Usage rule | This component is typically used as a single-component sub-job but can also be used as output component. |

**Putting files onto an FTP server**

Here is an example of using Talend FTP components to put several files in a local directory onto an FTP server.

### Creating a Job for putting files onto an FTP server

Create a Job to connect to an FTP server, then put several local files onto the server, finally close the connection to the server.
Procedure

1. Create a new Job and add a **tFTPConnection** component, a **tFTPPut** component, and a **tFTPClose** component by typing their names in the design workspace or dropping them from the **Palette**.
2. Link the **tFTPConnection** component to the **tFTPPut** component using a **Trigger > OnSubjobOk** connection.
3. Link the **tFTPPut** component to the **tFTPClose** component using a **Trigger > OnSubjobOk** connection.

Opening a connection to the FTP server

Configure the **tFTPConnection** component to open a connection to the FTP server.

Procedure

1. Double-click the **tFTPConnection** component to open its **Basic settings** view.
2. In the **Host** and **Port** fields, enter the FTP server IP address and the listening port number respectively.
3. In the **Username** and **Password** fields, enter the authentication details.
4. From the **Connection Mode** drop-down list, select the FTP connection mode you want to use, **Active** in this example.

Putting files onto the FTP server

Configure the **tFTPPut** component to put several local files onto the FTP server root directory.

Procedure

1. Double-click the **tFTPPut** component to open its **Basic settings** view.
2. Specify the connection details required to access the FTP server. In this example, select the **Use an existing connection** check box and from the **Component list** drop-down list displayed, select the connection component to reuse its connection details you have already defined.

3. In the **Local directory** field, specify the local directory that contains the files to be put onto the FTP server. In this example, it is `D:/components`.

4. In the **Remote directory** field, specify the FTP server directory onto which the files will be put. In this example, it is `/`, which means the root directory of the FTP server.

5. Clear the **Move to the current directory** check box.

6. In the **Files** table, click twice the `[+]` button to add two lines, and in the two **Filemask** column fields, enter `*.txt` and `*.png` respectively, which means only the text and png files in the specified local directory will be put onto the FTP server root directory.

**Closing the connection to the FTP server**

Configure the **tFTPClose** component to close the connection to the FTP server.

**Procedure**

1. Double-click the **tFTPClose** component to open its **Basic settings** view.

2. From the **Component list** drop-down list, select the **tFTPConnection** component that opens the connection you need to close. In this example, only one **tFTPConnection** component is used and it is selected by default.
Executing the Job to put files on the FTP server

After setting up the Job and configuring the components used in the Job for putting files onto the FTP server, you can then execute the Job and verify the Job execution result.

Procedure

1. Press Ctrl + S to save the Job and then F6 to execute the Job.
2. Connect to the FTP server to verify the result.

As shown above, only the text and png files in the local directory are put onto the FTP server.
tFTPRename

Renames files in an FTP directory.

**tFTPRename Standard properties**

These properties are used to configure tFTPRename running in the Standard Job framework.

The Standard tFTPRename component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>Host</td>
<td>The IP address or hostname of the FTP server.</td>
</tr>
<tr>
<td>Port</td>
<td>The listening port number of the FTP server.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>The user authentication data to access the FTP server. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Remote directory</td>
<td>The path to the FTP directory where the files to be renamed are available.</td>
</tr>
<tr>
<td>Move to the current directory</td>
<td>Select this check box to change the directory to the one specified in the Remote directory field. The next FTP component in the Job will take this directory as the root of the remote directory when using the same connection.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the Use an existing connection check box is selected.</td>
</tr>
<tr>
<td>Overwrite file</td>
<td>Select the action to be performed when the file already exists.</td>
</tr>
</tbody>
</table>
- **never**: Never overwrite the file.
- **always**: Always overwrite the file.
- **size different**: Overwrite the file when the file size is different.

### SFTP Support
Select this check box to connect to the FTP server via an SFTP connection.

### Authentication method
Select the SFTP authentication method, either **Public key** or **Password**.
- **Public key**: Enter the path to the private key and the passphrase for the key in the **Private key** and **Key Passphrase** fields correspondingly.
- **Password**: Enter the password required.
This property is available only when the SFTP Support check box is selected.

### Filename encoding
Select this check box to set the encoding used to convert file names from Strings to bytes. It should be the same encoding used on the SFTP server. If the SFTP server's version is greater than 3, the encoding should be **UTF-8**, or else an error occurs.
This property is available only when the SFTP Support check box is selected.

### FTPS Support
Select this check box to connect to the FTP server via an FTPS connection.

### Keystore File
The path to your keystore file, a password protected file containing several keys and certificates.
This property is available only when the FTPS Support check box is selected.

### Keystore Password
The password for your keystore file.
This property is available only when the FTPS Support check box is selected.

### Security Mode
Select the security mode from the list, either **Implicit** or **Explicit**.
This property is available only when the FTPS Support check box is selected.

### Files
Specify the files to be renamed and their new names.
- **Filemask**: specify the file to be renamed by entering the filename or filemask using wildcharacters or regular expressions.
- **New name**: enter the new name of the file.

### Connection mode
Select the connection mode from the list, either **Passive** or **Active**.
### Encoding
Specify the encoding type by selecting an encoding type from the list or selecting **CUSTOM** and enter the encoding type manually.

### Die on error
Select the check box to stop the execution of the Job when an error occurs.
Clear the check box to skip any error and continue the Job execution process.

#### Advanced settings

<table>
<thead>
<tr>
<th>Use Socks Proxy</th>
<th>Select this check box if you are using a proxy, and in the <strong>Proxy host</strong>, <strong>Proxy port</strong>, <strong>Proxy user</strong> and <strong>Proxy password</strong> fields displayed, specify the proxy server settings information.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignore Failure At Quit (FTP)</td>
<td>Select this check box to ignore library closing errors or FTP closing errors.</td>
</tr>
<tr>
<td>Data Channel Protection Level</td>
<td>The data channel protection level with which data is transferred between the client and the server. For more information, see RFC 2228: FTP Security Extensions. This property is available only when the <strong>FTPS Support</strong> check box is selected.</td>
</tr>
<tr>
<td>Protection Buffer Size</td>
<td>The maximum size (in bytes) of the encoded data blocks to be transferred between the client and the server. For more information, see RFC 2228: FTP Security Extensions. This property is available only when the <strong>FTPS Support</strong> check box is selected.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

#### Global Variables

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_FILE</td>
<td>The number of the files processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>CURRENT_STATUS</td>
<td>The execution result of the component. This is a Flow variable and it returns a string.</td>
</tr>
</tbody>
</table>

### Usage

| Usage rule | This component is generally used as a subjob with one component, but it can also be used as an output or end component. |
Renaming a file located on an FTP server

Here is an example of using Talend FTP components to rename a file located on an FTP server.

Creating a Job for renaming a file on an FTP server

Create a Job to connect to an FTP server, then rename a file on the server, finally close the connection to the server.

Before you begin

Prerequisites: To replicate this scenario, an FTP server must be started and a file must be put onto the server. In this example, the file movies.json has been put into the folder movies under the root directory of the FTP server.

Procedure

1. Create a new Job and add a tFTPConnection component, a tFTPRename component, and a tFTPClose component by typing their names in the design workspace or dropping them from the Palette.

2. Link the tFTPConnection component to the tFTPRename component using a Trigger > OnSubjobOk connection.

3. Link the tFTPRename component to the tFTPClose component using a Trigger > OnSubjobOk connection.
Opening a connection to the FTP server

Configure the tFTPConnection component to open a connection to the FTP server.

Procedure

1. Double-click the tFTPConnection component to open its Basic settings view.
2. In the Host and Port fields, enter the FTP server IP address and the listening port number respectively.
3. In the Username and Password fields, enter the authentication details.

Renaming the file on the FTP server

Configure the tFTPRename component to rename the file on the FTP server.

Procedure

1. Double-click the tFTPRename component to open its Basic settings view.
2. Specify the connection details required to access the FTP server. In this example, select the Use an existing connection check box and from the Component list drop-down list displayed, select the connection component to reuse its connection details you have already defined.
3. In the Remote directory field, enter the directory on the FTP server where the file to be renamed exists. In this example, it is /movies.
4. Clear the Move to the current directory check box.
5. In the Files table, click the [+ ] button to add a line, and then enter the existing file name in the Filemask column field and the new file name in the New name column field. In this example, they are movies.json and action_movies.json respectively.

Closing the connection to the FTP server

Configure the tFTPClose component to close the connection to the FTP server.

Procedure

1. Double-click the tFTPClose component to open its Basic settings view.
2. From the **Component list** drop-down list, select the **tFTPConnection** component that opens the connection you need to close. In this example, only one **tFTPConnection** component is used and it is selected by default.

**Executing the Job to rename the file on the FTP server**

After setting up the Job and configuring the components used in the Job for renaming the file on the FTP server, you can then execute the Job and verify the Job execution result.

**Procedure**

1. Press **Ctrl + S** to save the Job and then **F6** to execute the Job.
2. Connect to the FTP server to verify the result.

As shown above, the file on the FTP server has been renamed from `movies.json` to `action_movies.json`.
# tFTPTruncate

Truncates files in an FTP directory.

## tFTPTruncate Standard properties

These properties are used to configure tFTPTruncate running in the Standard Job framework.

The Standard tFTPTruncate component belongs to the Internet family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>The IP address or hostname of the FTP server.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>The listening port number of the FTP server.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>The user authentication data to access the FTP server. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Remote directory</strong></td>
<td>The path to the FTP directory in which the files will be truncated.</td>
</tr>
<tr>
<td><strong>Move to the current directory</strong></td>
<td>Select this check box to change the directory to the one specified in the Remote directory field. The next FTP component in the Job will take this directory as the root of the remote directory when using the same connection. This property is available only when the Use an existing connection check box is selected.</td>
</tr>
<tr>
<td><strong>SFTP Support</strong></td>
<td>Select this check box to connect to the FTP server via an SFTP connection.</td>
</tr>
</tbody>
</table>
| Authentication method | Select the SFTP authentication method, either **Public key** or **Password**.  
|                       | • **Public key**: Enter the path to the private key and the passphrase for the key in the **Private key** and **Key Passphrase** fields correspondingly.  
|                       | • **Password**: Enter the password required.  
|                       | This property is available only when the **SFTP Support** check box is selected. |
| Filename encoding     | Select this check box to set the encoding used to convert file names from Strings to bytes. It should be the same encoding used on the SFTP server. If the SFTP server’s version is greater than 3, the encoding should be **UTF-8**, or else an error occurs.  
|                       | This property is available only when the **SFTP Support** check box is selected. |
| FTPS Support          | Select this check box to connect to the FTP server via an FTPS connection. |
| Keystore File         | The path to your keystore file, a password protected file containing several keys and certificates.  
|                       | This property is available only when the **FTPS Support** check box is selected. |
| Keystore Password     | The password for your keystore file.  
|                       | This property is available only when the **FTPS Support** check box is selected. |
| Security Mode         | Select the security mode from the list, either **Implicit** or **Explicit**.  
|                       | This property is available only when the **FTPS Support** check box is selected. |
| Use Perl5 Regex Expression as Filemask | Select this check box to use Perl5 regular expressions in the **Files** field as file filters. This is useful when the name of the file to be processed contains special characters such as parentheses.  
|                       | For more information about Perl5 regular expression syntax, see **Perl5 Regular Expression Syntax**. |
| Files                 | The names of the files or the paths to the files to be truncated. You can specify multiple files in a line by using wildcards or a regular expression. |
| Connection mode       | Select the connection mode from the list, either **Passive** or **Active**. |
| Encoding              | Specify the encoding type by selecting an encoding type from the list or selecting **CUSTOM** and enter the encoding type manually. |
Advanced settings

| Use Socks Proxy | Select this check box if you are using a proxy, and in the **Proxy host**, **Proxy port**, **Proxy user** and **Proxy password** fields displayed, specify the proxy server settings information. |
| Ignore Failure At Quit (FTP) | Select this check box to ignore library closing errors or FTP closing errors. |
| Data Channel Protection Level | The data channel protection level with which data is transferred between the client and the server. For more information, see RFC 2228: FTP Security Extensions. This property is available only when the **FTPS Support** check box is selected. |
| Protection Buffer Size | The maximum size (in bytes) of the encoded data blocks to be transferred between the client and the server. For more information, see RFC 2228: FTP Security Extensions. This property is available only when the **FTPS Support** check box is selected. |
| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

Global Variables

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |
| NB_FILE | The number of the files processed. This is an After variable and it returns an integer. |
| CURRENT_STATUS | The execution result of the component. This is a Flow variable and it returns a string. |

Usage

| Usage rule | This component is typically used as a single-component sub-job but can also be used with other components. |

Related scenario

No scenario is available for this component yet.
tFuzzyMatch

Compares a column from the main flow with a reference column from the lookup flow and outputs the main flow data displaying the distance.

**tFuzzyMatch Standard properties**

These properties are used to configure tFuzzyMatch running in the Standard Job framework.

The Standard tFuzzyMatch component belongs to the Data Quality family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit schema | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository.
Two read-only columns, Value and Match are added to the output schema automatically. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: The schema already exists and is stored in the Repository, hence can be reused in various projects and job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>
| Matching type | Select the relevant matching algorithm among:
**Levenshtein**: Based on the edit distance theory. It calculates the number of insertion, deletion or substitution required for an entry to match the reference entry.  
**Metaphone**: Based on a phonetic algorithm for indexing entries by their pronunciation. It first loads the phonetics of all entries of the lookup reference and checks all entries of the main flow against the entries of the reference flow.  
**Double Metaphone**: a new version of the Metaphone phonetic algorithm, that produces more accurate results than the original algorithm. It can return both a primary and a secondary code for a string. This accounts for some ambiguous cases as well as for multiple variants of surnames with common ancestry. |
| Min distance | (Levenshtein only) Set the minimum number of changes allowed to match the reference. If set to 0, only perfect matches are returned. |
### Max distance
(Levenshtein only) Set the maximum number of changes allowed to match the reference.

### Matching column
Select the column of the main flow that needs to be checked against the reference (lookup) key column.

### Unique matching
Select this check box if you want to get the best match possible, in case several matches are available.

### Matching item separator
In case several matches are available, all of them are displayed unless the unique match box is selected. Define the delimiter between all matches.

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

| Usage rule | This component is not startable (green background) and it requires two input components and an output component. |

### Scenario 1: Levenshtein distance of 0 in first names

This scenario describes a four-component Job aiming at checking the edit distance between the First Name column of an input file with the data of the reference input file. The output of this Levenshtein type check is displayed along with the content of the main flow on a table.
Setting up the Job

Procedure
1. Drag and drop the following components from the Palette to the design workspace: tFileInputDelimited (x2), tFuzzyMatch, tLogRow.
2. Link the first tFileInputDelimited component to the tFuzzyMatch component using a Row > Main connection.
3. Link the second tFileInputDelimited component to the tFuzzyMatch using a Row > Main connection (which appears as a Lookup row on the design workspace).
4. Link the tFuzzyMatch component to the standard output tLogRow using a Row > Main connection.

Configuring the components

Procedure
1. Define the first tFileInputDelimited in its Basic settings view. Browse the system to the input file to be analyzed.
2. Define the schema of the component. In this example, the input schema has two columns, firstname and gender.
3. Define the second tFileInputDelimited component the same way.
   
   Warning:
   Make sure the reference column is set as key column in the schema of the lookup flow.

4. Double-click the tFuzzyMatch component to open its Basic settings view, and check its schema.
The **Schema** should match the **Main** input flow schema in order for the main flow to be checked against the reference.

Note that two columns, **Value** and **Matching**, are added to the output schema. These are standard matching information and are read-only.

5. Select the method to be used to check the incoming data. In this scenario, **Levenshtein** is the **Matching type** to be used.

6. Then set the distance. In this method, the distance is the number of char changes (insertion, deletion or substitution) that needs to be carried out in order for the entry to fully match the reference.

In this use case, we set both the minimum distance and the maximum distance to 0. This means only the exact matches will be output.

7. Also, clear the **Case sensitive** check box.

8. Check that the matching column and look up column are correctly selected.

9. Leave the other parameters as default.

**Executing the Job**

**Procedure**

Save the Job and press **F6** to execute the Job.
Results

As the edit distance has been set to 0 (min and max), the output shows the result of a regular join between the main flow and the lookup (reference) flow, hence only full matches with Value of 0 are displayed.

A more obvious example is with a minimum distance of 1 and a maximum distance of 2, see Procedure on page 1265

Scenario 2: Levenshtein distance of 1 or 2 in first names

This scenario is based on the scenario described above. Only the minimum and maximum distance settings in the tFuzzyMatch component are modified, which will change the output displayed.

Procedure

Procedure

1. In the Component view of the tFuzzyMatch, change the minimum distance from 0 to 1. This excludes straight away the exact matches (which would show a distance of 0).
2. Change also the maximum distance to 2. The output will provide all matching entries showing a discrepancy of 2 characters at most.

No other changes are required.
3. Make sure the **Matching item separator** is defined, as several references might be matching the main flow entry.

4. Save the new Job and press **F6** to run it.

As the edit distance has been set to 2, some entries of the main flow match more than one reference entry.

**Results**

You can also use another method, the metaphone, to assess the distance between the main flow and the reference, which will be described in the next scenario.

**Scenario 3: Metaphonic distance in first name**

This scenario is based on the scenario described above.

**Procedure**

1. Change the **Matching type** to **Metaphone**. There is no minimum nor maximum distance to set as the matching method is based on the discrepancies with the phonetics of the reference.

2. Save the Job and press **F6**. The phonetics value is displayed along with the possible matches.
<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>First Initial</th>
<th>Last Initial</th>
<th>Full Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew</td>
<td>M</td>
<td>ANTR</td>
<td>Andrews</td>
<td>Andrews</td>
</tr>
<tr>
<td>Thomas</td>
<td>M</td>
<td>ANTR</td>
<td>MS</td>
<td>Thomas</td>
</tr>
<tr>
<td>Ronald</td>
<td>M</td>
<td>ANTR</td>
<td>Ronald</td>
<td>Ronald</td>
</tr>
<tr>
<td>Milla</td>
<td>F</td>
<td>XLI</td>
<td>Mille</td>
<td>Milla</td>
</tr>
<tr>
<td>Richard</td>
<td>M</td>
<td>ANTR</td>
<td>Richard</td>
<td>Richard</td>
</tr>
<tr>
<td>Jack</td>
<td>M</td>
<td>XLI</td>
<td>Jack</td>
<td>Jack</td>
</tr>
<tr>
<td>Hellen</td>
<td>F</td>
<td>ANTR</td>
<td>Hellen</td>
<td>Hellen</td>
</tr>
<tr>
<td>Roger</td>
<td>M</td>
<td>RJR</td>
<td>Roger</td>
<td>Roger</td>
</tr>
<tr>
<td>Roberto</td>
<td>M</td>
<td>ANTR</td>
<td>Roberto</td>
<td>Roberto</td>
</tr>
<tr>
<td>Martin</td>
<td>M</td>
<td>ANTR</td>
<td>Martin</td>
<td>Martin</td>
</tr>
<tr>
<td>Lisa</td>
<td>F</td>
<td>LSI</td>
<td>Lisa</td>
<td>Lisa</td>
</tr>
<tr>
<td>Martin</td>
<td>M</td>
<td>ANTR</td>
<td>Martin</td>
<td>Martin</td>
</tr>
<tr>
<td>William</td>
<td>M</td>
<td>WLI</td>
<td>William</td>
<td>William</td>
</tr>
<tr>
<td>George</td>
<td>M</td>
<td>JRJ</td>
<td>George</td>
<td>George</td>
</tr>
<tr>
<td>Tom</td>
<td>M</td>
<td>TM</td>
<td>Tommy</td>
<td>Tom</td>
</tr>
<tr>
<td>John</td>
<td>M</td>
<td>JN</td>
<td>Johnny, John</td>
<td>John, John</td>
</tr>
</tbody>
</table>
tGoogleDataprocManage

Creates or deletes a Dataproc cluster on Google Cloud Platform.

**tGoogleDataprocManage Standard properties**

These properties are used to configure tGoogleDataprocManage running in the Standard Job framework.

The Standard tGoogleDataprocManage component belongs to the Cloud family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project identifier</td>
<td>Enter the ID of your Google Cloud Platform project. If you are not certain about your project ID, check it in the Manage Resources page of your Google Cloud Platform services.</td>
</tr>
<tr>
<td>Cluster identifier</td>
<td>Enter the ID of your Dataproc cluster to be used.</td>
</tr>
<tr>
<td>Provide Google Credentials in file</td>
<td>Leave this check box clear, when you launch your Job from a given machine in which Google Cloud SDK has been installed and authorized to use your user account credentials to access Google Cloud Platform. In this situation, this machine is often your local machine. For further information about this Google Credentials file, see the administrator of your Google Cloud Platform or visit Google Cloud Platform Auth Guide.</td>
</tr>
</tbody>
</table>
| Action                          | Select the action you want tGoogleDataprocManage to perform on the your cluster:  
  - Start to create a cluster  
  - Stop to destroy a cluster |
| Version                         | Select the version of the image to be used to create a Dataproc cluster. |
| Zone                            | Select the geographic location in which the computing resources are used and your data is stored and processed. |
| Instance configuration          | Enter the parameters to determine how many masters and workers to be used by the Dataproc cluster to be created and the performance of these masters and workers. |
### Advanced settings

<table>
<thead>
<tr>
<th><strong>Wait for cluster ready</strong></th>
<th>Select this check box to keep this component running until the cluster is completely set up. When you clear this check box, this component stops running immediately after sending the command to create.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Master disk size</strong></td>
<td>Enter a number without quotation marks to determine the size of the disk of each master instance.</td>
</tr>
<tr>
<td><strong>Master local SSD</strong></td>
<td>Enter a number without quotation marks to determine the number of local solid-state drive (SSD) storage devices to be added to each master instance. According to Google, these local SSDs are suitable only for temporary storage such as caches, processing space or low value data. It is recommended to store important data to durable storage options of Google. For further information about the Google storage options, see Durable storage options.</td>
</tr>
<tr>
<td><strong>Worker disk size</strong></td>
<td>Enter a number without quotation marks to determine the size of the disk of each worker instance.</td>
</tr>
<tr>
<td><strong>Worker local SSD</strong></td>
<td>Enter a number without quotation marks to determine the number of local solid-state drive (SSD) storage devices to be added to each worker instance. According to Google, these local SSDs are suitable only for temporary storage such as caches, processing space or low value data. It is recommended to store important data to durable storage options of Google. For further information about the Google storage options, see Durable storage options.</td>
</tr>
<tr>
<td><strong>Network or Subnetwork</strong></td>
<td>Select either check box to use a Google Compute Engine network or subnetwork for the cluster to be created to enable intra-cluster communications. As Google does not allow network and subnetwork to be used concurrently, selecting one check box hides the other check box. For further information about Google Dataproc cluster network configuration, see Dataproc Network.</td>
</tr>
</tbody>
</table>

**Initialization action**

In this table, select the initialization actions that are available in the shared bucket on Google Cloud Storage to run on all the nodes in your Dataproc cluster immediately after this cluster is set up.

If you need to use custom initialization scripts, upload them to this shared Google bucket so that tGoogleDataprocManage can read them.

- In the **Executable file** column, enter the Google Cloud Storage URI to these scripts to be used, for example `gs://dataproc-initialization-actions/MyScript`
- In the **Executable timeout** column, enter the amount of time within double quotation marks to determine the duration of the execution. If the executable is not completed at the end of this timeout, an explanatory error message is returned. The value is a string with up to nine fractional digits, for example, “3.5s” for 3.5 seconds.

For further information about this shared bucket and the initialization actions, see Initialization actions.

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

### Usage

| **Usage rule** | This component is used standalone in a Subjob. |
tGoogleDriveConnection

Opens a Google Drive connection that can be reused by other Google Drive components.

**tGoogleDriveConnection Standard properties**

These properties are used to configure tGoogleDriveConnection running in the Standard Job framework.

The Standard tGoogleDriveConnection component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application Name</th>
<th>The application name required by Google Drive to get access to its APIs.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>OAuth Method</th>
<th>Select an OAuth method used to access Google Drive from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Access Token (deprecated)</strong>: uses an access token to access Google Drive.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Installed Application (Id &amp; Secret)</strong>: uses the client ID and client secret created through Google API Console to access Google Drive. For more information about this method, see Google Identity Platform &gt; Installed applications.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Installed Application (JSON)</strong>: uses the client secret JSON file that is created through Google API Console and contains the client ID, client secret, and other OAuth 2.0 parameters to access Google Drive.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Service Account</strong>: uses a service account JSON file created through Google API Console to access Google Drive. For more information about this method, see Google Identity Platform &gt; Service accounts.</td>
</tr>
</tbody>
</table>

For more detailed information about how to access Google Drive using each method, see OAuth methods for accessing Google Drive on page 1272.
| **Access Token** | The access token generated through Google Developers OAuth 2.0 Playground. This property is available only when **Access Token** is selected from the **OAuth Method** drop-down list. |
| **Client ID** and **Client Secret** | The client ID and client secret. These two properties are available only when **Installed Application (Id & Secret)** is selected from the **OAuth Method** drop-down list. |
| **Client Secret JSON** | The path to the client secret JSON file. This property is available only when **Installed Application (JSON)** is selected from the **OAuth Method** drop-down list. |
| **Service Account JSON** | The path to the service account JSON file. This property is available only when **Service Account** is selected from the **OAuth Method** drop-down list. |
| **Use Proxy** | Select this check box when you are working behind a proxy. With this check box selected, you need to specify the value for the following parameters:  
  - **Host**: The IP address of the HTTP proxy server.  
  - **Port**: The port number of the HTTP proxy server. |
| **Use SSL** | Select this check box if an SSL connection is used to access Google Drive. With this check box selected, you need to specify the value for the following parameters:  
  - **Algorithm**: The name of the SSL cryptography algorithm.  
  - **Keystore File**: The path to the certificate TrustStore file that contains the list of certificates the client trusts.  
  - **Password**: The password used to check the integrity of the TrustStore data. |

**Advanced settings**

| **DataStore Path** | The path to the credential file that stores the refresh token.  
  
  **Note**: When your client ID, client secret, or any other configuration related to the Installed Application authentication changes, you need to delete this credential file manually before running your Job again.  
  This property is available only when **Installed Application (Id & Secret)** or **Installed Application (JSON)** is selected from the **OAuth Method** drop-down list. |
tGoogleDriveConnection

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

Global Variables

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other Google Drive components. In a Job design, it is usually used to open a Google Drive connection that can be reused by other Google Drive components.</th>
</tr>
</thead>
</table>

OAuth methods for accessing Google Drive

Talend provides the following four OAuth methods to access Google Drive using Google Drive components and metadata wizard.

- Installed Application (Id & Secret)
- Installed Application (JSON)
- Service Account
- Access Token (deprecated)

How to access Google Drive using client ID and secret

To use client ID and client secret to access Google Drive, you need to first generate the client ID and client secret by completing the following steps using Google Chrome.

Before you begin

A Google account has already been signed up for using Google Drive.

Procedure

1. Go to Google API Console and select an existing project or create a new one. In this example, we create a new project TalendProject.
2. Go to the Library page and in the right panel, find Google Drive API and enable the Google Drive API that allows you to access resources from Google Drive.
3. Go to the Credentials page, click OAuth consent screen in the right panel and set a product name in the Product name shown to users field. In this example, it is TalendProduct. When done, click Save.
4. Click **Create credentials > OAuth client ID**, and in the **Create client ID** page, create a new client ID **TalendApplication** with **Application type** set to **Other**.
5. Click **Create**. You will be shown your client ID and client secret that can be used by Google Drive components and metadata wizard to access Google Drive using the OAuth method **Installed Application (Id & Secret)**.

---

**How to access Google Drive using a client secret JSON file**

To use a client secret JSON file to access Google Drive, you need to first download the client secret JSON file from Google API Console by completing the following steps using Google Chrome.
Before you begin

The client ID and client secret have been created in Google API Console. For more information, see How to access Google Drive using client ID and secret on page 1272.

Procedure

1. Go to Google API Console.
2. Go to the Credentials page.
3. Click the Download JSON button to download the client secret JSON file and securely store it in a local folder. This JSON file can then be used by Google Drive components and metadata wizard to access Google Drive via the OAuth method Installed Application (JSON).

How to access Google Drive using a service account JSON file

To use a service account JSON file to access Google Drive, you need to first create a service account in Google API Console, then download the service account JSON file by completing the following steps using Google Chrome.

Before you begin

1. A Google account has already been signed up for using Google Drive.
2. In Google API Console, your project has been created, the Google Drive API has been enabled, and the product name has been set. For more information about how to make these configuration, see How to access Google Drive using client ID and secret on page 1272.

Procedure

1. Go to Google API Console.
2. Open the Service accounts page. If prompted, select your project.
3. Click **CREATE SERVICE ACCOUNT**.
4. In the **Create service account** window, type a name for the service account, select **Furnish a new private key** and then the key type **JSON**.
5. Click Create. In the pop-up window, choose a folder and click Save to store your service account JSON file securely. This JSON file can then be used by Google Drive components and metadata wizard to access Google Drive via the OAuth method Service Account.

How to access Google Drive using an access token (deprecated)

To use an access token to access Google Drive, you need to first generate the access token by completing the following steps using Google Developers OAuth Playground.

Before you begin

1. A Google account has already been signed up for using Google Drive.

2. The client ID and client secret have been created in Google API Console. For more information, see How to access Google Drive using client ID and secret on page 1272.

Procedure

1. Go to Google Developers OAuth Playground.

2. Click OAuth 2.0 Configuration and select Use your own OAuth credentials check box, enter the OAuth client ID and client secret you have already created in the OAuth Client ID and OAuth Client secret fields respectively.
3. In OAuth 2.0 Playground Step 1, select the scope `https://www.googleapis.com/auth/drive` under Drive API v3 for the Google Drive API and click Authorize APIs, then click Allow to generate the authorization code.
4. In OAuth 2.0 Playground Step 2, click **Exchange authorization code for tokens** to generate the OAuth access token.

The OAuth access token is displayed on the right panel as shown in below figure. It can be used by Google Drive components and metadata wizard to access Google Drive via the OAuth method **Access Token**.
Note that the access token expires in every 3600 seconds. You can click **Refresh access token** in OAuth 2.0 Playground Step 2 to refresh it.

**Related scenario**

Managing files with Google Drive on page 1304
**tGoogleDriveCopy**

Creates a copy of a file/folder in Google Drive.

**tGoogleDriveCopy Standard properties**

These properties are used to configure tGoogleDriveCopy running in the Standard Job framework.

The Standard tGoogleDriveCopy component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

This property is not available when other connection component is selected from the Connection Component drop-down list.

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component that opens the database connection to be reused by this component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Name</td>
<td>The application name required by Google Drive to get access to its APIs.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OAuth Method</th>
<th>Select an OAuth method used to access Google Drive from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Access Token (deprecated):</strong> uses an access token to access Google Drive.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Installed Application (Id &amp; Secret):</strong> uses the client ID and client secret created through Google API Console to access Google Drive. For more information about this method, see Google Identity Platform &gt; Installed applications.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Installed Application (JSON):</strong> uses the client secret JSON file that is created through Google API Console and contains the client ID, client secret, and other OAuth 2.0 parameters to access Google Drive.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Service Account:</strong> uses a service account JSON file created through Google API Console to access Google Drive. For more information about this</td>
</tr>
</tbody>
</table>
For more detailed information about how to access Google Drive using each method, see OAuth methods for accessing Google Drive on page 1272.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Token</strong></td>
<td>The access token generated through Google Developers OAuth 2.0 Playground.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when Access Token is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Client ID and Client Secret</strong></td>
<td>The client ID and client secret.</td>
</tr>
<tr>
<td></td>
<td>These two properties are available only when Installed Application (Id &amp; Secret) is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Client Secret JSON</strong></td>
<td>The path to the client secret JSON file.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when Installed Application (JSON) is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Service Account JSON</strong></td>
<td>The path to the service account JSON file.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when Service Account is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Use Proxy</strong></td>
<td>Select this check box when you are working behind a proxy. With this check box selected, you need to specify the value for the following parameters:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Host</strong>: The IP address of the HTTP proxy server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Port</strong>: The port number of the HTTP proxy server.</td>
</tr>
<tr>
<td><strong>Use SSL</strong></td>
<td>Select this check box if an SSL connection is used to access Google Drive.</td>
</tr>
<tr>
<td></td>
<td>With this check box selected, you need to specify the value for the following parameters:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Algorithm</strong>: The name of the SSL cryptography algorithm.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Keystore File</strong>: The path to the certificate TrustStore file that contains the list of certificates the client trusts.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Password</strong>: The password used to check the integrity of the TrustStore data.</td>
</tr>
<tr>
<td><strong>Copy Mode</strong></td>
<td>Select the type of the item to be copied.</td>
</tr>
<tr>
<td></td>
<td>• <strong>File</strong>: Select this option when you need to copy a file.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Folder</strong>: Select this option when you need to copy a folder.</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>The name or ID of the source file/folder to be copied.</td>
</tr>
<tr>
<td><strong>Source Access Mode</strong></td>
<td>Select the method by which the source file/folder is specified, either by Name or by Id.</td>
</tr>
<tr>
<td><strong>Destination Folder Name</strong></td>
<td>The name or ID of the destination folder in which the copy of the source file/folder will be saved.</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Destination Access Mode</strong></td>
<td>Select the method by which the destination folder is specified, either by Name or by Id.</td>
</tr>
<tr>
<td><strong>Rename (set new title)</strong></td>
<td>Select this check box to rename the copy of the file/folder in the destination folder. In the Destination Name field displayed, enter the name for the file/folder after being copied to the destination folder.</td>
</tr>
<tr>
<td><strong>Remove Source File</strong></td>
<td>Select this check box to remove the source file after it is copied to the destination folder. This property is available only when File is selected from the Copy Mode drop-down list.</td>
</tr>
</tbody>
</table>
| **Schema and Edit schema** | A schema is a row description, and it defines the fields to be processed and passed on to the next component. The schema of this component is read-only. You can click the [...] button next to Edit schema to view the predefined schema which contains the following fields:  
  - sourceId: The ID of the source file/folder.  
  - destinationId: The ID of the destination file/folder. |

**Advanced settings**

| **DataStore Path** | The path to the credential file that stores the refresh token.  
  Note: When your client ID, client secret, or any other configuration related to the Installed Application authentication changes, you need to delete this credential file manually before running your Job again.  
  This property is available only when Installed Application (Id & Secret) or Installed Application (JSON) is selected from the OAuth Method drop-down list. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th><strong>ERROR_MESSAGE</strong></th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SOURCE_ID</strong></td>
<td>The ID of the source file/folder. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td><strong>DESTINATION_ID</strong></td>
<td>The ID of the destination file/folder. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>
**Usage**

| Usage rule | This component can be used as a standalone component or as a start component of a Job or Subjob. |

**Related scenario**

Managing files with Google Drive on page 1304
**tGoogleDriveCreate**

Creates a new folder in Google Drive.

### tGoogleDriveCreate Standard properties

These properties are used to configure tGoogleDriveCreate running in the Standard Job framework.

The Standard tGoogleDriveCreate component belongs to the Cloud family.

The component in this framework is generally available.

#### Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
<tr>
<td><strong>Connection Component</strong></td>
<td>Select the component that opens the database connection to be reused by this component.</td>
</tr>
<tr>
<td><strong>Application Name</strong></td>
<td>The application name required by Google Drive to get access to its APIs.</td>
</tr>
<tr>
<td><strong>OAuth Method</strong></td>
<td>Select an OAuth method used to access Google Drive from the drop-down list.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Access Token (deprecated)</strong>: uses an access token to access Google Drive.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Installed Application (Id &amp; Secret)</strong>: uses the client ID and client secret created through Google API Console to access Google Drive. For more information about this method, see <a href="#">Google Identity Platform &gt; Installed applications</a>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Installed Application (JSON)</strong>: uses the client secret JSON file that is created through Google API Console and contains the client ID, client secret, and other OAuth 2.0 parameters to access Google Drive.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Service Account</strong>: uses a service account JSON file created through Google API Console to access Google Drive. For more information about this method, see <a href="#">Google Identity Platform &gt; Service accounts</a>.</td>
</tr>
</tbody>
</table>
For more detailed information about how to access Google Drive using each method, see OAuth methods for accessing Google Drive on page 1272.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Required Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Token</strong></td>
<td>The access token generated through Google Developers OAuth 2.0 Playground.</td>
<td>This property is available only when Access Token is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Client ID and Client Secret</strong></td>
<td>The client ID and client secret.</td>
<td>These two properties are available only when Installed Application (Id &amp; Secret) is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Client Secret JSON</strong></td>
<td>The path to the client secret JSON file.</td>
<td>This property is available only when Installed Application (JSON) is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Service Account JSON</strong></td>
<td>The path to the service account JSON file.</td>
<td>This property is available only when Service Account is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Use Proxy</strong></td>
<td>Select this check box when you are working behind a proxy. With this check box selected, you need to specify the value for the following parameters:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Host</strong>: The IP address of the HTTP proxy server.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Port</strong>: The port number of the HTTP proxy server.</td>
<td></td>
</tr>
<tr>
<td><strong>Use SSL</strong></td>
<td>Select this check box if an SSL connection is used to access Google Drive.</td>
<td>With this check box selected, you need to specify the value for the following parameters:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Algorithm</strong>: The name of the SSL cryptography algorithm.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Keystore File</strong>: The path to the certificate TrustStore file that contains the list of certificates the client trusts.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Password</strong>: The password used to check the integrity of the TrustStore data.</td>
<td></td>
</tr>
<tr>
<td><strong>Parent Folder</strong></td>
<td>The name or ID of the parent folder in which a new folder will be created.</td>
<td></td>
</tr>
<tr>
<td><strong>Access Method</strong></td>
<td>Select the method by which the parent folder is specified, either by Name or by Id.</td>
<td></td>
</tr>
<tr>
<td><strong>New Folder Name</strong></td>
<td>The name of the new folder to be created.</td>
<td></td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description, and it defines the fields to be processed and passed on to the next component.</td>
<td></td>
</tr>
</tbody>
</table>
The schema of this component is read-only. You can click the [...] button next to Edit schema to view the predefined schema which contains the following fields:

- **parentFolderId**: the ID of the parent folder.
- **newFolderId**: the ID of the new folder.

---

### Advanced settings

<table>
<thead>
<tr>
<th>DataStore Path</th>
<th>The path to the credential file that stores the refresh token.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note</strong>: When your client ID, client secret, or any other configuration related to the Installed Application authentication changes, you need to delete this credential file manually before running your Job again.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when Installed Application (Id &amp; Secret) or Installed Application (JSON) is selected from the OAuth Method drop-down list.</td>
</tr>
</tbody>
</table>

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

### Global Variables

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARENT_FOLDER_ID</td>
<td>The ID of the parent folder. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>NEW_FOLDER_ID</td>
<td>The ID of the new folder. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

---

### Usage

| Usage rule | This component can be used as a standalone component or as a start component of a Job or Subjob. |

### Related scenario

[Managing files with Google Drive](#) on page 1304
tGoogleDriveDelete

Deletes a file/folder in Google Drive.

**tGoogleDriveDelete Standard properties**

These properties are used to configure tGoogleDriveDelete running in the Standard Job framework.

The Standard tGoogleDriveDelete component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

This property is not available when other connection component is selected from the Connection Component drop-down list.

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component that opens the database connection to be reused by this component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Name</td>
<td>The application name required by Google Drive to get access to its APIs.</td>
</tr>
<tr>
<td>OAuth Method</td>
<td>Select an OAuth method used to access Google Drive from the drop-down list.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Access Token (deprecated)</strong>: uses an access token to access Google Drive.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Installed Application (Id &amp; Secret)</strong>: uses the client ID and client secret created through Google API Console to access Google Drive. For more information about this method, see Google Identity Platform &gt; Installed applications.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Installed Application (JSON)</strong>: uses the client secret JSON file that is created through Google API Console and contains the client ID, client secret, and other OAuth 2.0 parameters to access Google Drive.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Service Account</strong>: uses a service account JSON file created through Google API Console to access Google Drive. For more information about this</td>
</tr>
</tbody>
</table>
For more detailed information about how to access Google Drive using each method, see OAuth methods for accessing Google Drive on page 1272.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Token</strong></td>
<td>The access token generated through Google Developers OAuth 2.0 Playground.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when Access Token is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Client ID and Client Secret</strong></td>
<td>The client ID and client secret.</td>
</tr>
<tr>
<td></td>
<td>These two properties are available only when Installed Application (Id &amp; Secret) is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Client Secret JSON</strong></td>
<td>The path to the client secret JSON file.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when Installed Application (JSON) is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Service Account JSON</strong></td>
<td>The path to the service account JSON file.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when Service Account is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Use Proxy</strong></td>
<td>Select this check box when you are working behind a proxy. With this check box selected, you need to specify the value for the following parameters:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Host</strong>: The IP address of the HTTP proxy server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Port</strong>: The port number of the HTTP proxy server.</td>
</tr>
<tr>
<td><strong>Use SSL</strong></td>
<td>Select this check box if an SSL connection is used to access Google Drive. With this check box selected, you need to specify the value for the following parameters:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Algorithm</strong>: The name of the SSL cryptography algorithm.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Keystore File</strong>: The path to the certificate TrustStore file that contains the list of certificates the client trusts.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Password</strong>: The password used to check the integrity of the TrustStore data.</td>
</tr>
<tr>
<td><strong>File/Folder</strong></td>
<td>The name or ID of the file/folder to be deleted.</td>
</tr>
<tr>
<td><strong>Delete Mode</strong></td>
<td>Select the method by which the file/folder to be deleted is specified, either by Name or by Id.</td>
</tr>
<tr>
<td><strong>Use Trash</strong></td>
<td>Select this check box to move the file/folder to be deleted to the trash.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description, and it defines the fields to be processed and passed on to the next component.</td>
</tr>
</tbody>
</table>
The schema of this component is read-only. You can click the [...] button next to Edit schema to view the predefined schema with only one field named fileId which describes the ID of the file/folder.

**Advanced settings**

| DataStore Path | The path to the credential file that stores the refresh token.  
Note: When your client ID, client secret, or any other configuration related to the Installed Application authentication changes, you need to delete this credential file manually before running your Job again.  
This property is available only when Installed Application (Id & Secret) or Installed Application (JSON) is selected from the OAuth Method drop-down list. |
|----------------|----------------------------------------------------------------------------------------------------------|

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_ID</td>
<td>The ID of the file/folder. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as a standalone component or as a start component of a Job or Subjob.</th>
</tr>
</thead>
</table>

**Related scenario**

No scenario is available for this component yet.
tGoogleDriveGet

Gets a file’s content and downloads the file to a local directory.

**tGoogleDriveGet Standard properties**

These properties are used to configure tGoogleDriveGet running in the Standard Job framework.

The Standard tGoogleDriveGet component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in. This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
</tbody>
</table>

| Connection Component | Select the component that opens the database connection to be reused by this component. |
| Application Name | The application name required by Google Drive to get access to its APIs. |
| OAuth Method | Select an OAuth method used to access Google Drive from the drop-down list. |
| |  - **Access Token (deprecated)**: uses an access token to access Google Drive. |
| |  - **Installed Application (Id & Secret)**: uses the client ID and client secret created through Google API Console to access Google Drive. For more information about this method, see Google Identity Platform > Installed applications. |
| |  - **Installed Application (JSON)**: uses the client secret JSON file that is created through Google API Console and contains the client ID, client secret, and other OAuth 2.0 parameters to access Google Drive. |
| |  - **Service Account**: uses a service account JSON file created through Google API Console to access Google Drive. For more information about this |
For more detailed information about how to access Google Drive using each method, see OAuth methods for accessing Google Drive on page 1272.

<table>
<thead>
<tr>
<th>Access Token</th>
<th>The access token generated through Google Developers OAuth 2.0 Playground. This property is available only when Access Token is selected from the OAuth Method drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client ID and Client Secret</td>
<td>The client ID and client secret. These two properties are available only when Installed Application (Id &amp; Secret) is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td>Client Secret JSON</td>
<td>The path to the client secret JSON file. This property is available only when Installed Application (JSON) is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td>Service Account JSON</td>
<td>The path to the service account JSON file. This property is available only when Service Account is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td>Use Proxy</td>
<td>Select this check box when you are working behind a proxy. With this check box selected, you need to specify the value for the following parameters: • <strong>Host</strong>: The IP address of the HTTP proxy server. • <strong>Port</strong>: The port number of the HTTP proxy server.</td>
</tr>
<tr>
<td>Use SSL</td>
<td>Select this check box if an SSL connection is used to access Google Drive. With this check box selected, you need to specify the value for the following parameters: • <strong>Algorithm</strong>: The name of the SSL cryptography algorithm. • <strong>Keystore File</strong>: The path to the certificate TrustStore file that contains the list of certificates the client trusts. • <strong>Password</strong>: The password used to check the integrity of the TrustStore data.</td>
</tr>
<tr>
<td>File</td>
<td>The name or ID of the file to be downloaded.</td>
</tr>
<tr>
<td>Access Method</td>
<td>Select the method by which the file to be downloaded is specified, either by Name or by Id.</td>
</tr>
<tr>
<td>Save as File</td>
<td>Select this check box to save the file to a local directory. In the Save to field displayed, browse to or enter the path where you want to save the file to be downloaded.</td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description, and it defines the fields to be processed and passed on to the next component.</td>
</tr>
</tbody>
</table>
The schema of this component is read-only. You can click the [...] button next to **Edit schema** to view the predefined schema with only one field named **content** which describes the content of the file to be downloaded.

### Advanced settings

**DataStore Path**

The path to the credential file that stores the refresh token.

**Note:** When your client ID, client secret, or any other configuration related to the Installed Application authentication changes, you need to delete this credential file manually before running your Job again.

This property is available only when **Installed Application (Id & Secret)** or **Installed Application (JSON)** is selected from the **OAuth Method** drop-down list.

<table>
<thead>
<tr>
<th>Export Google Doc as</th>
<th>Select the type for the Google Doc to be exported.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export Google Draw as</td>
<td>Select the type for the Google Draw to be exported.</td>
</tr>
<tr>
<td>Export Google Presentation as</td>
<td>Select the type for the Google Presentation to be exported.</td>
</tr>
<tr>
<td>Export Google Spreadsheet as</td>
<td>Select the type for the Google Spreadsheet to be exported.</td>
</tr>
<tr>
<td>Add extension</td>
<td>Select this check box to add extension to the exported file.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_ID</td>
<td>The ID of the file. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

### Usage

| Usage rule | This component can be used as a standalone component or as a start component of a Job or Subjob. |
Related scenario

No scenario is available for this component yet.
tGoogleDriveList

Lists all files, or folders, or both files and folders in a specified Google Drive folder.

**tGoogleDriveList Standard properties**

These properties are used to configure tGoogleDriveList running in the Standard Job framework.

The Standard tGoogleDriveList component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td>Connection Component</td>
<td>Select the component that opens the database connection to be reused by this component.</td>
</tr>
<tr>
<td>Application Name</td>
<td>The application name required by Google Drive to get access to its APIs.</td>
</tr>
<tr>
<td>OAuth Method</td>
<td>Select an OAuth method used to access Google Drive from the drop-down list.</td>
</tr>
</tbody>
</table>

- **Built-In**: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.

- **Repository**: The connection details stored centrally in Repository > Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.

This property is not available when other connection component is selected from the Connection Component drop-down list.

- **OAuth Method**
  - **Access Token (deprecated)**: uses an access token to access Google Drive.
  - **Installed Application (Id & Secret)**: uses the client ID and client secret created through Google API Console to access Google Drive. For more information about this method, see Google Identity Platform > Installed applications.
  - **Installed Application (JSON)**: uses the client secret JSON file that is created through Google API Console and contains the client ID, client secret, and other OAuth 2.0 parameters to access Google Drive.
  - **Service Account**: uses a service account JSON file created through Google API Console to access Google Drive. For more information about this...
For more detailed information about how to access Google Drive using each method, see OAuth methods for accessing Google Drive on page 1272.

<table>
<thead>
<tr>
<th>Access Token</th>
<th>The access token generated through Google Developers OAuth 2.0 Playground. This property is available only when Access Token is selected from the OAuth Method drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client ID and Client Secret</td>
<td>The client ID and client secret. These two properties are available only when Installed Application (Id &amp; Secret) is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td>Client Secret JSON</td>
<td>The path to the client secret JSON file. This property is available only when Installed Application (JSON) is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td>Service Account JSON</td>
<td>The path to the service account JSON file. This property is available only when Service Account is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td>Use Proxy</td>
<td>Select this check box when you are working behind a proxy. With this check box selected, you need to specify the value for the following parameters:</td>
</tr>
<tr>
<td></td>
<td>• Host: The IP address of the HTTP proxy server.</td>
</tr>
<tr>
<td></td>
<td>• Port: The port number of the HTTP proxy server.</td>
</tr>
<tr>
<td>Use SSL</td>
<td>Select this check box if an SSL connection is used to access Google Drive. With this check box selected, you need to specify the value for the following parameters:</td>
</tr>
<tr>
<td></td>
<td>• Algorithm: The name of the SSL cryptography algorithm.</td>
</tr>
<tr>
<td></td>
<td>• Keystore File: The path to the certificate TrustStore file that contains the list of certificates the client trusts.</td>
</tr>
<tr>
<td></td>
<td>• Password: The password used to check the integrity of the TrustStore data.</td>
</tr>
<tr>
<td>Folder Name</td>
<td>The name or ID of the folder in which the files/folders will be listed.</td>
</tr>
<tr>
<td>Access Method</td>
<td>Select the method by which the folder is specified, either by Name or by Id.</td>
</tr>
<tr>
<td>FileList Type</td>
<td>Select the type of data you want to list.</td>
</tr>
<tr>
<td></td>
<td>• Files: Only files.</td>
</tr>
<tr>
<td></td>
<td>• Directories: Only folders.</td>
</tr>
</tbody>
</table>
### Include SubDirectories

Select this check box to list also the files/folders in the subdirectories.

### Schema and Edit schema

A schema is a row description, and it defines the fields to be processed and passed on to the next component.

The schema of this component is read-only. You can click the [...] button next to Edit schema to view the predefined schema which contains the following fields:

- **id**: The ID of the file/folder.
- **name**: The name of the file/folder.
- **mimeType**: The MIME type of the file/folder.
- **modifiedTime**: The last modification date of the file/folder.
- **size**: The file size in bytes.
- **kind**: The kind of the resource.
- **trashed**: Whether the file has been trashed.
- **parents**: The ID of the parent folder.
- **webViewLink**: A link for opening the file in a Google editor or viewer in a browser.

### Advanced settings

#### DataStore Path

The path to the credential file that stores the refresh token.

**Note**: When your client ID, client secret, or any other configuration related to the Installed Application authentication changes, you need to delete this credential file manually before running your Job again.

This property is available only when Installed Application (Id & Secret) or Installed Application (JSON) is selected from the OAuth Method drop-down list.

#### Include trashed files

Select this check box to also take into account files and folders that have been removed from the specified path.

#### tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

#### ERROR_MESSAGE

The error message generated by the component when an error occurs. This is an After variable and it returns a string.
Usage

| Usage rule | This component is usually used as a start component of a Job or Subjob and it always needs an output link. |

Related scenario

Managing files with Google Drive on page 1304
tGoogleDrivePut

Uploads data from a data flow or a local file to Google Drive.

**tGoogleDrivePut Standard properties**

These properties are used to configure tGoogleDrivePut running in the Standard Job framework.

The Standard tGoogleDrivePut component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td><strong>Connection Component</strong></td>
<td>Select the component that opens the database connection to be reused by this component.</td>
</tr>
<tr>
<td><strong>Application Name</strong></td>
<td>The application name required by Google Drive to get access to its APIs.</td>
</tr>
<tr>
<td><strong>OAuth Method</strong></td>
<td>Select an OAuth method used to access Google Drive from the drop-down list.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Access Token (deprecated)</strong>: uses an access token to access Google Drive.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Installed Application (Id &amp; Secret)</strong>: uses the client ID and client secret created through Google API Console to access Google Drive. For more information about this method, see Google Identity Platform &gt; Installed applications .</td>
</tr>
<tr>
<td></td>
<td>• <strong>Installed Application (JSON)</strong>: uses the client secret JSON file that is created through Google API Console and contains the client ID, client secret, and other OAuth 2.0 parameters to access Google Drive.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Service Account</strong>: uses a service account JSON file created through Google API Console to access Google Drive. For more information about this</td>
</tr>
</tbody>
</table>
method, see Google Identity Platform > Service accounts.

For more detailed information about how to access Google Drive using each method, see OAuth methods for accessing Google Drive on page 1272.

<table>
<thead>
<tr>
<th><strong>Access Token</strong></th>
<th>The access token generated through Google Developers OAuth 2.0 Playground. This property is available only when Access Token is selected from the OAuth Method drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client ID and Client Secret</strong></td>
<td>The client ID and client secret. These two properties are available only when Installed Application (Id &amp; Secret) is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Client Secret JSON</strong></td>
<td>The path to the client secret JSON file. This property is available only when Installed Application (JSON) is selected from the OAuth Method drop-down list.</td>
</tr>
<tr>
<td><strong>Service Account JSON</strong></td>
<td>The path to the service account JSON file. This property is available only when Service Account is selected from the OAuth Method drop-down list.</td>
</tr>
</tbody>
</table>
| **Use Proxy** | Select this check box when you are working behind a proxy. With this check box selected, you need to specify the value for the following parameters:  
  - **Host**: The IP address of the HTTP proxy server.  
  - **Port**: The port number of the HTTP proxy server. |
| **Use SSL** | Select this check box if an SSL connection is used to access Google Drive. With this check box selected, you need to specify the value for the following parameters:  
  - **Algorithm**: The name of the SSL cryptography algorithm.  
  - **Keystore File**: The path to the certificate TrustStore file that contains the list of certificates the client trusts.  
  - **Password**: The password used to check the integrity of the TrustStore data. |
| **File Name** | The name for the file after being uploaded. |
| **Destination Folder** | The name or ID of the folder in which uploaded data will be stored. |
| **Access Method** | Select the method by which the destination folder is specified, either by Name or by Id. |
| **Replace if Existing** | Select this check box to overwrite any existing file with the newly uploaded one. |
### Upload Mode

Select one of the following upload modes from the drop-down list:

- **Upload Incoming content as File**: Select this option to upload data from an input flow of the preceding component.
- **Upload Local File**: Select this option to upload data from a local file. In the File field displayed, specify the path to the file to be uploaded.
- **Expose As OutputStream**: Select this option to expose output stream of this component, which can be used by other components to write the file content. For example, you can use the Use Output Stream feature of the tFileOutputDelimited component to feed a given tGoogleDrivePut's exposed output stream. For more information, see tFileOutputDelimited on page 1110.

### Schema and Edit schema

A schema is a row description, and it defines the fields to be processed and passed on to the next component. The schema of this component is read-only. You can click the [...] button next to Edit schema to view the predefined schema which contains the following fields:

- **content**: The content of the uploaded data.
- **parentFolderId**: The ID of the parent folder.
- **fileId**: The ID of the file.

### Advanced settings

**DataStore Path**

The path to the credential file that stores the refresh token.

**Note**: When your client ID, client secret, or any other configuration related to the Installed Application authentication changes, you need to delete this credential file manually before running your Job again.

This property is available only when Installed Application (Id & Secret) or Installed Application (JSON) is selected from the OAuth Method drop-down list.

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**ERROR_MESSAGE**

The error message generated by the component when an error occurs. This is an After variable and it returns a string.
### Usage

**Usage rule**

This component can be used as a standalone component to upload a local file to Google Drive or as an end component to upload data from an input flow of the preceding component to Google Drive.

---

### Managing files with Google Drive

This scenario describes a Job that uploads two files to an empty folder *Talend* in the root directory of Google Drive, then creates a new folder *Talend Backup* in the root directory and copies one of the two files to the new folder *Talend Backup*, and finally lists and displays all files and folders in the root directory of Google Drive on the console.
Creating a Job for managing files with Google Drive

Procedure

1. Create a new Job and add a `tGoogleDriveConnection` component, two `tGoogleDrivePut` components, a `tFileInputRaw` component, a `tGoogleDriveCreate` component, a `tGoogleDriveCopy` component, a `tGoogleDriveList` component, and five `tLogRow` components to the Job.
2. Link the first `tGoogleDrivePut` component to the first `tLogRow` component using a `Row > Main` connection.

3. Do the same to link the `tFileInputRaw` component to the second `tGoogleDrivePut` component, the second `tGoogleDrivePut` component to the second `tLogRow` component, the `tGoogleDriveCreate` component to the third `tLogRow` component, the `tGoogleDriveCopy` component to the fourth `tLogRow` component, the `tGoogleDriveList` component to the fifth `tLogRow` component.

4. Link the `tGoogleDriveConnection` component to the first `tGoogleDrivePut` component using a `Trigger > On Subjob Ok` connection.

5. Do the same to link the first `tGoogleDrivePut` component to the `tFileInputRaw` component, the `tFileInputRaw` component to the `tGoogleDriveCreate` component, the `tGoogleDriveCreate` component to the `tGoogleDriveCopy` component, and the `tGoogleDriveCopy` component to the `tGoogleDriveList` component.

### Opening a connection to Google Drive

Configure the `tGoogleDriveConnection` component to connect to Google Drive using a client secret JSON file.
Before you begin

- The client secret JSON file has been downloaded into a local folder through Google API Console. For more information, see How to access Google Drive using a client secret JSON file on page 1276.
- An empty folder Talend has been created in the root directory of Google Drive.

Procedure

1. Double-click the tGoogleDriveConnection component to open its Basic settings view in the Component tab.

2. In the Application Name field, enter the application name required by Google Drive to get access to its API. In this example, it is TalendProject.

3. Select Installed Application (JSON) from the OAuth Method drop-down list.

4. In the Client Secret JSON field, specify the path to the client secret JSON file you have generated, D:/client_secret.json in this example.

Uploading files to Google Drive

Procedure

1. Double-click the first tGoogleDrivePut component to open its Basic settings view in the Component tab.
2. Select the component that will create the Google Drive connection from the **Connection Component** drop-down list, **tGoogleDriveConnection_1** in this example.

3. Select by Name from the **Access Method** drop-down list and in the **Destination Folder** field, enter the name of the folder in which the file will be uploaded, **Talend** in this example.

   **Note:** When accessing a Google Drive resource by its name, if the name matches more than one resource, an error will be thrown because the resource cannot be identified precisely. In this case, you can specify the Google Drive resource using a pseudo path hierarchy, like /Talend/Documentation. This example specifies a folder named Documentation under the folder Talend under the Google Drive root folder.

4. In the **File Name** field, enter the name for the file after being uploaded. In this example, it is Talend Customers.csv.

5. Select Upload Local File from the **Upload Mode** drop-down list and in the **File** field, browse to or enter the path to the file to be uploaded. In this example, it is D:/Downloads/Talend Customers.csv.

6. Double-click the **tFileInputRaw** component and on its **Basic settings** view, select **Read the file as a bytes array** in the **Mode** area and specify the path to the file whose content will be uploaded in the **Filename** field, D:/Downloads/Talend Products.txt in this example.

7. Double-click the second **tGoogleDrivePut** component to open its **Basic settings** view in the **Component** tab.

8. Repeat step 2 on page 1308 to step 3 on page 1308 to configure this component.

9. In the **File Name** field, enter the name for the file after being uploaded. In this example, it is Talend Products.txt.

10. Select Upload Incoming content as File from the **Upload Mode** drop-down list.
Creating a new folder in Google Drive

Procedure
1. Double-click tGoogleDriveCreate to open its Basic settings view in the Component tab.
   - Connection Component: tGoogleDriveConnection_1
   - Parent Folder: root
   - New Folder Name: Talend Backup

2. Select the component that will create the Google Drive connection from the Connection Component drop-down list, tGoogleDriveConnection_1 in this example.
3. In the Parent Folder field, enter the name of the folder in which a new folder will be created. In this example, it is root.
4. In the New Folder Name field, enter the name of the folder to be created. In this example, it is Talend Backup.
5. Double-click the third tLogRow component to open its Basic settings view in the Component tab.
6. In the Mode area, select Vertical (each row is a key/value list) for a better display of the results.

Copying a file to the newly created folder

Procedure
1. Double-click the tGoogleDriveCopy component to open its Basic settings view in the Component tab.
   - Connection Component: tGoogleDriveConnection_1
   - Copy Mode: File
   - Source: Talend Customers.csv
   - Destination Folder Name: Talend Backup
   - Destination Name: Talend Customers v1.0.csv

2. Select the component that will create the Google Drive connection from the Connection Component drop-down list, tGoogleDriveConnection_1 in this example.
3. Select File from the Copy Mode drop-down list.
4. In the Source field, enter the name of the file to be copied. In this example, it is Talend Customers.csv.
5. In the Destination Folder Name field, enter the name of the folder to which the file will be copied. In this example, it is Talend Backup.
6. Select the Rename (set new title) check box and in the Destination Name field, enter a new name for the file after being copied to the destination folder. In this example, it is Talend Customers v1.0.csv.
7. Double-click the fourth `tLogRow` component to open its **Basic settings** view in the **Component** tab.

8. In the **Mode** area, select **Vertical (each row is a key/value list)** for a better display of the results.

### Listing files and folders in Google Drive

**Procedure**

1. Double-click the `tGoogleDriveList` component to open its **Basic settings** view in the **Component** tab.

   ![tGoogleDriveList](image)

   - **Connection Component**: `tGoogleDriveConnection_1`
   - **Folder Name**: `root`
   - **FileList Type**: `Both`
   - **Include SubDirectories**
   - **Schema**: `Built-In`

2. Select the component that will create the Google Drive connection from the **Connection Component** drop-down list, `tGoogleDriveConnection_1` in this example.

3. In the **Folder Name** field, enter the name of the folder in which the files/folders will be listed. In this example, it is the root directory of Google Drive and you can use the alias `root` to refer to it.

4. Select **Both** from the **FileList Type** drop-down list to list both files and folders in the root directory.

5. Select the **Include SubDirectories** check box to list also the files/folders in the subdirectories.

6. Double-click the fifth `tLogRow` component to open its **Basic settings** view in the **Component** tab.

   ![tLogRow](image)

   - **Mode**: **Vertical (each row is a key/value list)**

7. In the **Mode** area, select **Vertical (each row is a key/value list)** for a better display of the results.

### Saving and executing the Job

**Procedure**

1. Press **Ctrl + S** to save the Job.

2. Execute the Job by pressing **F6** or clicking **Run** on the **Run** tab.
<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>parentFolderId</td>
<td>0B2ZtkTEM5ALpaXZ5eQpMfQ4MxKX1</td>
</tr>
<tr>
<td>fileId</td>
<td>1afs7VT8U_RW0UMrY6Qj_655vVx88MjvwG6</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>#1. tLogRow_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
</tr>
<tr>
<td>content</td>
</tr>
<tr>
<td>1:Accolade 2:Color 6s 3:Johnson Controls</td>
</tr>
<tr>
<td>parentFolderId</td>
</tr>
<tr>
<td>fileId</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>#1. tLogRow_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
</tr>
<tr>
<td>content</td>
</tr>
<tr>
<td>parentFolderId</td>
</tr>
<tr>
<td>fileId</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>#1. tLogRow_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
</tr>
<tr>
<td>parentFolderId</td>
</tr>
<tr>
<td>newFolderId</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>#1. tLogRow_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
</tr>
<tr>
<td>sourceId</td>
</tr>
<tr>
<td>destinationId</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>#1. tLogRow_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
</tr>
<tr>
<td>id</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>mimeType</td>
</tr>
<tr>
<td>size</td>
</tr>
<tr>
<td>kind</td>
</tr>
<tr>
<td>trashed</td>
</tr>
<tr>
<td>parents</td>
</tr>
</tbody>
</table>
As shown above, two files *Talend Products.txt* and *Talend Customers.csv* were uploaded to the folder *Talend*, then a new folder *Talend Backup* was created in the root folder and the file *Talend Customers.csv* was copied to the new folder and renamed to *Talend Customers v1.0.csv*, and finally all files and folders in the root directory are listed on the console.
tGPGDecrypt

Calls the `gpg -d` command to decrypt a GnuPG-encrypted file and saves the decrypted file in the specified directory.

**tGPGDecrypt Standard properties**

These properties are used to configure tGPGDecrypt running in the Standard Job framework.
The Standard tGPGDecrypt component belongs to the File family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Input encrypted file</th>
<th>File path to the encrypted file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output decrypted file</td>
<td>File path to the output decrypted file.</td>
</tr>
<tr>
<td>GPG binary path</td>
<td>File path to the GPG command.</td>
</tr>
<tr>
<td>Passphrase</td>
<td>Enter the passphrase used in encrypting the specified input file.</td>
</tr>
<tr>
<td></td>
<td>To enter the passphrase, click the [...] button next to the passphrase field, and then in the pop-up dialog box enter the passphrase between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>No TTY Terminal</td>
<td>Select this check box to specify that no TTY terminal is used by adding the --no-tty option to the decryption command.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | FILE: the name of the output file. This is a Flow variable and it returns a string. |
|------------------| FILEPATH: the path of the output file. This is a Flow variable and it returns a string. |
|                  | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |
|                  | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as a standalone component.</th>
</tr>
</thead>
</table>

Scenario: Decrypt a GnuPG-encrypted file and display its content

The following scenario describes a three-component Job that decrypts a GnuPG-encrypted file and displays the content of the decrypted file on the Run console.

Dragging and linking the components

Procedure

1. Drop a tGPGDecrypt component, a tFileInputDelimited component, and a tLogRow component from the Palette to the design workspace.
2. Connect the tGPGDecrypt component to the tFileInputDelimited component using a Trigger > OnSubjobOk link, and connect the tFileInputDelimited component to the tLogRow component using a Row > Main link.

Configuring the components

Procedure

1. Double-click the tGPGDecrypt to open its Component view and set its properties:
2. In the **Input encrypted file** field, browse to the file to be decrypted.
3. In the **Output decrypted file** field, enter the path to the decrypted file.

   **Warning:**
   If the file path contains accented characters, you will get an error message when running the Job.

4. In the **GPG binary path** field, browse to the GPG command file.
5. In the **Passphrase** field, enter the passphrase used when encrypting the input file.
6. Double-click the **tFileInputDelimited** component to open its **Component** view and set its properties:

   7. In the **File name/Stream** field, define the path to the decrypted file, which is the output path you have defined in the **tGPGDecrypt** component.
8. In the **Header**, **Footer** and **Limit** fields, define respectively the number of rows to be skipped in the beginning of the file, at the end of the file and the number of rows to be processed.
9. Use a **Built-In** schema. This means that it is available for this Job only.
10. Click **Edit schema** and edit the schema for the component. Click twice the [+ ] button to add two columns that you will call **idState** and **labelState**.
11. Click **OK** to validate your changes and close the editor.
12. Double-click the **tLogRow** component and set its properties:

13. Use a **Built-In** schema for this scenario.

14. In the **Mode** area, define the console display mode according to your preference. In this scenario, select **Table (print values in cells of a table)**.

### Saving and executing the Job

**Procedure**

1. Press **Ctrl+S** to save your Job
2. Press **F6** or click **Run** from the **Run** tab to run it.
**Results**

The specified file is decrypted and the defined number of rows of the decrypted file are printed on the **Run** console.
tGreenplumBulkExec

Improves performance when loading data in a Greenplum database.

The tGreenplumOutputBulk and tGreenplumBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT statement used to feed a database. These two steps are fused together in the tGreenplumOutputBulkExec component, detailed in a separate section. The advantage of using a two step process is that it makes it possible to transform data before it is loaded in the database.

tGreenplumBulkExec performs an Insert action on the data.

tGreenplumBulkExec Standard properties

These properties are used to configure tGreenplumBulkExec running in the Standard Job framework. The Standard tGreenplumBulkExec component belongs to the Databases family.

The component in this framework is generally available.

i Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

i Note:
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

| **Host** | Database server IP address. |
| **Port** | Listening port number of DB server. |
| **Database** | Name of the database. |
| **Schema** | Exact name of the schema. |
| **Username and Password** | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Table** | Name of the table to be written. Note that only one table can be written at a time |
| **Action on table** | On the table defined, you can perform one of the following operations: **None**: No operation is carried out. **Drop and create a table**: The table is removed and created again. **Create a table**: The table does not exist and gets created. **Create a table if not exists**: The table is created if it does not exist. **Drop a table if exists and create**: The table is removed if it already exists and created again. **Clear a table**: The table content is deleted. |
| **Filename** | Name of the file to be loaded. **Warning**: This file is located on the machine specified by the URI in the Host field so it should be on the same machine as the database server. |
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. **Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*. **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various
projects and Job designs. Related topic: see Talend Studio User Guide.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

• **View schema:** choose this option to view the schema only.

• **Change to built-in property:** choose this option to change the schema to Built-in for local changes.

• **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

### Advanced settings

<table>
<thead>
<tr>
<th>Action on data</th>
<th>Select the operation you want to perform: Bulk insert, Bulk update. The details asked will be different according to the action chosen.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy the OID for each row</td>
<td>Retrieve the ID item for each row.</td>
</tr>
<tr>
<td>Contains a header line with the names of each column in the file</td>
<td>Specify that the table contains header.</td>
</tr>
<tr>
<td>File type</td>
<td>Select the file type to process.</td>
</tr>
<tr>
<td>Null string</td>
<td>String displayed to indicate that the value is null.</td>
</tr>
<tr>
<td>Fields terminated by</td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td>Escape char</td>
<td>Character of the row to be escaped.</td>
</tr>
<tr>
<td>Text enclosure</td>
<td>Character used to enclose text.</td>
</tr>
<tr>
<td>Force not null for columns</td>
<td>Define the columns nullability</td>
</tr>
<tr>
<td>Force not null</td>
<td>Select this check box next to the column you want to define as not null.</td>
</tr>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is generally used with a <strong>tGreenplumOutputBulk</strong> component. Used together they offer gains in performance while feeding a Greenplum database.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.  

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |

### Related scenarios

For more information about **tGreenplumBulkExec**, see:

- Scenario: Inserting transformed data in MySQL database on page 2537.
- Scenario: Inserting data in bulk in MySQL database on page 2544.
- Scenario: Truncating and inserting file data into an Oracle database on page 2752.
tGreenplumClose

Closes a connection to the Greenplum database.

**tGreenplumClose Standard properties**

These properties are used to configure tGreenplumClose running in the Standard Job framework.

The Standard tGreenplumClose component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tGreenplumConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is to be used along with Greenplum components, especially with tGreenplumConnection and tGreenplumCommit. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context- |
Related scenarios

No scenario is available for the Standard version of this component yet.
tGreenplumCommit

Commits global transaction in one go instead of repeating the operation for every row or every batch and thus provides gain in performance.

tGreenplumCommit validates the data processed through the Job into the connected DB. This component uses an unique connection.

**tGreenplumCommit Standard properties**

These properties are used to configure tGreenplumCommit running in the Standard Job framework.

The Standard tGreenplumCommit component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tGreenplumConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Warning:**

*If you want to use a Row > Main connection to link tGreenplumCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.*

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tGreenplum* components, especially with the |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

### Related scenarios

For tGreenplumCommit related scenarios, see:

- Scenario: Mapping data using a simple implicit join on page 682.
- Scenario: Inserting data in mother/daughter tables on page 2476.
tGreenplumConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

tGreenplumConnection opens a connection to the database for a current transaction.

**tGreenplumConnection Standard properties**

These properties are used to configure tGreenplumConnection running in the Standard Job framework.

The Standard tGreenplumConnection component belongs to the Databases and the ELT families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see *Dynamic database components on page 598.*

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>

**Use or register a shared DB Connection**

Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the Shared DB Connection Name field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among
several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the Use dynamic job and Use an independent process to run subjob options of the tRunJob component. Using a shared connection together with a tRunJob component with either of these two options enabled will cause your Job to fail.

Advanced settings

<table>
<thead>
<tr>
<th>Auto Commit</th>
<th>Select this check box to commit any changes to the database automatically upon the transaction. With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component. Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule | This component is more commonly used with other tGreenplum* components, especially with the tGreenplumCommit and tGreenplumRollback components. |

Related scenarios

For tGreenplumConnection related scenarios, see:

- Scenario: Mapping data using a simple implicit join on page 682.
- tMysqlConnection on page 2475.
tGreenplumGPLoad

Bulk loads data into a Greenplum table either from an existing data file, an input flow, or directly from a data flow in streaming mode through a named-pipe.

`tGreenplumGPLoad` inserts data into a Greenplum database table using Greenplum's `gpload` utility.

**tGreenplumGPLoad Standard properties**

These properties are used to configure `tGreenplumGPLoad` running in the Standard Job framework. The Standard `tGreenplumGPLoad` component belongs to the Databases family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of the DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the Greenplum database.</td>
</tr>
<tr>
<td>Schema</td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table into which the data is to be inserted.</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations before loading the data:</td>
</tr>
<tr>
<td></td>
<td><strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear table</strong>: The table content is deleted before the data is loaded.</td>
</tr>
<tr>
<td></td>
<td><strong>Create table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td><strong>Create table if not exists</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create table</strong>: The table is removed and created again.</td>
</tr>
</tbody>
</table>
### Drop table if exists and create
The table is removed if it already exists and created again.

### Truncate table
The table content is deleted. You do not have the possibility to rollback the operation.

### Action on data
On the data of the table defined, you can perform:

- **Insert**: Add new entries to the table. If duplicates are found, Job stops.
- **Update**: Make changes to existing entries.
- **Merge**: Updates or adds data to the table.

**Warning:**

It is necessary to specify at least one column as a primary key on which the Update and Merge operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). To define the Update/Merge options, select in the Match Column column the check boxes corresponding to the column names that you want to use as a base for the Update and Merge operations, and select in the Update Column column the check boxes corresponding to the column names that you want to update. To define the Update condition, type in the condition that will be used to update the data.

### Schema and Edit schema
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](#).

- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](#).

Click **Edit schema** to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content](#) window.
## Data file

Full path to the data file to be used. If this component is used in standalone mode, this is the name of an existing data file to be loaded into the database. If this component is connected with an input flow, this is the name of the file to be generated and written with the incoming data to later be used with gpload to load into the database. This field is hidden when the **Use named-pipe** check box is selected.

## Use named-pipe

Select this check box to use a named-pipe. This option is only applicable when the component is connected with an input flow. When this check box is selected, no data file is generated and the data is transferred to gpload through a named-pipe. This option greatly improves performance in both Linux and Windows.

### Note:

This component on named-pipe mode uses a JNI interface to create and write to a named-pipe on any Windows platform. Therefore the path to the associated JNI DLL must be configured inside the Java library path. The component comes with two DLLs for both 32 and 64 bit operating systems that are automatically provided in the Studio with the component.

## Named-pipe name

Specify a name for the named-pipe to be used. Ensure that the name entered is valid.

## Die on error

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row > Rejects link.

## Advanced settings

### Use existing control file (YAML formatted)

Select this check box to provide a control file to be used with the gpload utility instead of specifying all the options explicitly in the component. When this check box is selected, **Data file** and the other gpload related options no longer apply. Refer to Greenplum’s gpload manual for details on creating a control file.

### Control file

Enter the path to the control file to be used, between double quotation marks, or click [...] and browse to the control file. This option is passed on to the gpload utility via the -f argument.

### CSV mode

Select this check box to include CSV specific parameters such as **Escape char** and **Text enclosure**.

### Field separator

Character, string, or regular expression used to separate fields.

### Warning:


| **Escape char** | Character of the row to be escaped. |
| **Text enclosure** | Character used to enclose text. |
| **Header (skips the first row of data file)** | Select this check box to skip the first row of the data file. |
| **Additional options** | Set the gpload arguments in the corresponding table. Click [+] as many times as required to add arguments to the table. Click the Parameter field and choose among the arguments from the list. Then click the corresponding Value field and enter a value between quotation marks. |
| **LOCAL_HOSTNAME**: The host name or IP address of the local machine on which gpload is running. If this machine is configured with multiple network interface cards (NICs), you can specify the host name or IP of each individual NIC to allow network traffic to use all NICs simultaneously. By default, the local machine’s primary host name or IP is used. |
| **PORT (gpfdist port)**: The specific port number that the gpfdist file distribution program should use. You can also specify a PORT_RANGE to select an available port from the specified range. If both PORT and PORT_RANGE are defined, then PORT takes precedence. If neither PORT or PORT_RANGE is defined, an available port between 8000 and 9000 is selected by default. If multiple host names are declared in LOCAL_HOSTNAME, this port number is used for all hosts. This configuration is desired if you want to use all NICs to load the same file or set of files in a given directory location. |
| **PORT_RANGE**: Can be used instead of PORT (gpfdist port) to specify a range of port numbers from which gpload can choose an available port for this instance of the gpfdist file distribution program. |
| **NULL_AS**: The string that represents a null value. The default is \N (backslash-N) in TEXT mode, and an empty value with no quotation marks in CSV mode. Any source data item that matches this string will be considered a null value. |
| **FORCE_NOT_NULL**: In CSV mode, processes each specified column as though it were quoted and hence not a NULL value. For the default null string in CSV mode (nothing between two delimiters), this causes missing values to be evaluated as zero-length strings. |
| **ERROR_LIMIT (2 or higher)**: Enables single row error isolation mode for this load operation. When enabled and the error limit count is not reached on any Greenplum segment instance during input processing,
all good rows will be loaded and input rows that have format errors will be discarded or logged to the table specified in \texttt{ERROR\_TABLE} if available. When the error limit is reached, input rows that have format errors will cause the load operation to abort. Note that single row error isolation only applies to data rows with format errors, for example, extra or missing attributes, attributes of a wrong data type, or invalid client encoding sequences. Constraint errors, such as primary key violations, will still cause the load operation to abort if encountered. When this option is not enabled, the load operation will abort on the first error encountered.

\textbf{ERROR\_TABLE}: When \texttt{ERROR\_LIMIT} is declared, specifies an error table where rows with formatting errors will be logged when running in single row error isolation mode. You can then examine this error table to see error rows that were not loaded (if any).

\textbf{Log file}: Browse to or enter the access path to the log file in your directory.

\textbf{Encoding}: Define the encoding type manually in the field.

\textbf{Specify gpload path}: Select this check box to specify the full path to the \texttt{gpload} executable. You must check this option if the gpload path is not specified in the \texttt{PATH} environment variable.

\textbf{Full path to gpload executable}: Full path to the \texttt{gpload} executable on the machine in use. It is advisable to specify the gpload path in the \texttt{PATH} environment variable instead of selecting this option.

\textbf{tStatCatcher Statistics}: Select this check box to collect log data at the component level.

\textbf{Global Variables}

\textbf{Global Variables}

\textbf{NB\_LINE}: the number of rows processed. This is an After variable and it returns an integer.

\textbf{GPLOAD\_OUTPUT}: the output information when the \texttt{gpload} utility is the executed. This is an After variable and it returns a string.

\textbf{ERROR\_MESSAGE}: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the \texttt{Die on error} check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press \texttt{Ctrl + Space} to access the variable list and choose the variable to use from it.
For further information about variables, see *Talend Studio User Guide*.

## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is mainly used when no particular transformation is required on the data to be loaded on to the database. This component can be used as a standalone or an output component.</th>
</tr>
</thead>
</table>

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the *Install* button on the *Component* tab view. You can also find out and add all missing JARs easily on the *Modules* tab in the *Integration* perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)). |

## Related scenario

For a related use case, see *Scenario: Inserting data in bulk in MySQL database* on page 2544.
**tGreenplumInput**

Reads a database and extracts fields based on a query.

tGreenplumInput executes a DB query with a strictly defined order which must correspond to the schema definition and then it passes on the field list to the next component via a Main row link.

**tGreenplumInput Standard properties**

These properties are used to configure tGreenplumInput running in the Standard Job framework.

The Standard tGreenplumInput component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>
### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

#### Built-In

You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](#).

#### Repository

You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](#).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content](#) window.

### Query type and Query

Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

### Guess Query

Click the **Guess Query** button to generate the query which corresponds to your table schema in the **Query** field.

### Guess schema

Click the **Guess schema** button to retrieve the table schema.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use cursor</strong></td>
<td>When selected, helps to decide the row set to work with at a time and thus optimize performance.</td>
</tr>
<tr>
<td><strong>Trim all the String/Char columns</strong></td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</td>
</tr>
<tr>
<td><strong>Trim column</strong></td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
Global Variables

**NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**

This component covers all possible SQL queries for Firebird databases.

**Dynamic settings**

Click the [+ ] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

Related scenarios

For related topics, see:
• **Scenario: Mapping data using a simple implicit join** on page 682.

See also related topic: **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489.
tGreenplumOutput

Executes the action defined on the table and/or on the data of a table, according to the input flow from the previous component.

tGreenplumOutput writes, updates, modifies or deletes the data in a database.

**tGreenplumOutput Standard properties**

These properties are used to configure tGreenplumOutput running in the Standard Job framework.

The Standard tGreenplumOutput component belongs to the Databases family.

The component in this framework is generally available.

<i>Note:</i>

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>![Database Connection Icon]</td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component <strong>Basic settings</strong> view.</td>
</tr>
<tr>
<td></td>
<td>For more information about setting up and storing database connection parameters, see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>![Database Connection Icon]</td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component</td>
</tr>
</tbody>
</table>
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
</tbody>
</table>
| Username and Password | DB user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Table           | Name of the table to be written. Note that only one table can be written at a time |
| Action on table | On the table defined, you can perform one of the following operations:
  - **None**: No operation is carried out.  
  - **Drop and create a table**: The table is removed and created again. 
  - **Create a table**: The table does not exist and gets created. 
  - **Create a table if not exists**: The table is created if it does not exist. 
  - **Drop a table if exists and create**: The table is removed if it already exists and created again. 
  - **Clear a table**: The table content is deleted. |
| Action on data  | On the data of the table defined, you can perform:
  - **Insert**: Add new entries to the table. If duplicates are found, Job stops. 
  - **Update**: Make changes to existing entries 
  - **Insert or update**: Insert a new record. If the record with the given reference already exists, an update would be made. 
  - **Update or insert**: Update the record with the given reference. If the record does not exist, a new record would be inserted. 
  - **Delete**: Remove entries corresponding to the input flow.  
  - **Warning**: It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking |
| **Schema and Edit Schema** | **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the **Update** and **Delete** operations. To do that: Select the **Use field options** check box and then in the **Key in update column**, select the check boxes next to the column names you want to use as a base for the **Update** operation. Do the same in the **Key in delete column** for the **Delete** operation. 

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word *line* when naming the fields. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In:</strong> You create and store the schema locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
<td><strong>Die on error</strong></td>
</tr>
<tr>
<td><strong>Repository:</strong> You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <strong>Talend Studio User Guide</strong>. When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
<td></td>
</tr>
</tbody>
</table>
### Advanced settings

<table>
<thead>
<tr>
<th>Commit every</th>
<th>Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional Columns</td>
<td>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</td>
</tr>
<tr>
<td>Name:</td>
<td>Type in the name of the schema column to be altered or inserted as new column</td>
</tr>
<tr>
<td>SQL expression:</td>
<td>Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td>Position:</td>
<td>Select Before, Replace or After following the action to be performed on the reference column.</td>
</tr>
<tr>
<td>Reference column:</td>
<td>Type in a column of reference that the tDBOutput can use to place or replace the new or altered column.</td>
</tr>
<tr>
<td>Use field options</td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td>Use Batch</td>
<td>Select this check box to activate the batch mode for data processing.</td>
</tr>
<tr>
<td>Note:</td>
<td>This check box is available only when you have selected the Insert, Update, or Delete option in the Action on data option.</td>
</tr>
<tr>
<td>Batch Size</td>
<td>Specify the number of records to be processed in each batch. This field appears only when the Use batch mode check box is selected.</td>
</tr>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_LINE_UPDATED: the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_INSERTED: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_DELETED: the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>
**NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

---

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible SQL queries for Greenplum databases. It allows you to carry out actions on a table or on the data of a table in a Greenplum database. It enables you to create a reject flow, with a Row &gt; Rejects link filtering the data in error. For a usage example, see Scenario: Retrieving data in error with a Reject link on page 2529.</th>
</tr>
</thead>
</table>

| Dynamic settings | Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the Component List box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*. |

---

### Related scenarios

For related scenarios, see:
• **Scenario: Mapping data using a simple implicit join** on page 682.

• **Scenario: Inserting a column and altering data using tMysqlOutput** on page 2521.
tGreenplumOutputBulk

Prepares the file to be used as parameter in the INSERT query to feed the Greenplum database. The tGreenplumOutputBulk and tGreenplumBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tGreenplumOutputBulkExec component, detailed in a separate section. The advantage of using a two step process is that it makes it possible to transform data before it is loaded in the database.

Writes a file with columns based on the defined delimiter and the Greenplum standards

### tGreenplumOutputBulk Standard properties

These properties are used to configure tGreenplumOutputBulk running in the Standard Job framework. The Standard tGreenplumOutputBulk component belongs to the Databases family. The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components on page 598](#).

### Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>File Name</strong></td>
<td>Name of the file to be generated. <strong>Warning:</strong> This file is generated on the local machine or a shared folder on the LAN.</td>
</tr>
<tr>
<td><strong>Append</strong></td>
<td>Select this check box to add the new rows at the end of the records</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
</tbody>
</table>
**Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](https://help.talend.com).

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](https://help.talend.com).

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-In** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row separator</strong></td>
<td>String (ex: &quot;\n&quot; on Unix) to distinguish rows.</td>
</tr>
<tr>
<td><strong>Field separator</strong></td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td><strong>Include header</strong></td>
<td>Select this check to include the column header.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td><strong>tStatCatcher statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong></td>
<td>the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions</td>
</tr>
</tbody>
</table>
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with <strong>tGreenplumBulkExec</strong> component. Used together they offer gains in performance while feeding a Greenplum database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component family</td>
<td>Databases/Greenplum</td>
</tr>
</tbody>
</table>

### Related scenarios

For use cases in relation with **tGreenplumOutputBulk**, see the following scenarios:

- **Scenario: Inserting transformed data in MySQL database** on page 2537.
- **Scenario: Inserting data in bulk in MySQL database** on page 2544.
**tGreenplumOutputBulkExec**

Provides performance gains during Insert operations to a Greenplum database.

The tGreenplumOutputBulk and tGreenplumBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tGreenplumOutputBulkExec component.

tGreenplumOutputBulkExec executes the action on the data provided.

**tGreenplumOutputBulkExec Standard properties**

These properties are used to configure tGreenplumOutputBulkExec running in the Standard Job framework.

The Standard tGreenplumOutputBulkExec component belongs to the Databases family.

The component in this framework is generally available.

<i>Note:</i>

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td></td>
<td>Currently, only localhost, 127.0.0.1 or the exact IP address of the local machine is allowed for proper functioning. In other words, the database server must be installed on the same machine where the Studio is installed or where the Job using tGreenplumOutputBulkExec is deployed.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database name</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>DB user authentication data.</td>
</tr>
</tbody>
</table>
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

<table>
<thead>
<tr>
<th>Table</th>
<th>Name of the table to be written. Note that only one table can be written at a time and that the table must exist for the insert operation to succeed.</th>
</tr>
</thead>
</table>
| Action on table | On the table defined, you can perform one of the following operations:  
**None**: No operation is carried out.  
**Drop and create a table**: The table is removed and created again.  
**Create a table**: The table does not exist and gets created.  
**Create a table if not exists**: The table is created if it does not exist.  
**Drop table if exists and create**: The table is removed if it already exists and created again.  
**Clear a table**: The table content is deleted. You have the possibility to rollback the operation. |
| File Name | Name of the file to be generated and loaded.  
**Warning**: This file is generated on the machine specified by the URI in the Host field so it should be on the same machine as the database server. |
| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
**Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.  
**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.  
When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.  
You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com). |
|  | Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available: |
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

## Advanced settings

<table>
<thead>
<tr>
<th>Action on data</th>
<th>Select the operation you want to perform:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bulk insert  Bulk update The details asked will be different according to the action chosen.</td>
</tr>
<tr>
<td>Copy the OID for each row</td>
<td>Retrieve the ID item for each row.</td>
</tr>
<tr>
<td>Contains a header line with the names of each column in the file</td>
<td>Specify that the table contains header.</td>
</tr>
<tr>
<td>File type</td>
<td>Select the file type to process.</td>
</tr>
<tr>
<td>Null string</td>
<td>String displayed to indicate that the value is null.</td>
</tr>
<tr>
<td>Row separator</td>
<td>String (ex: <code>\n</code>on Unix) to distinguish rows.</td>
</tr>
<tr>
<td>Fields terminated by</td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td>Escape char</td>
<td>Character of the row to be escaped</td>
</tr>
<tr>
<td>Text enclosure</td>
<td>Character used to enclose text.</td>
</tr>
<tr>
<td>Force not null for columns</td>
<td>Define the columns nullability</td>
</tr>
<tr>
<td></td>
<td><strong>Force not null</strong>: Select the check box next to the column you want to define as not null.</td>
</tr>
</tbody>
</table>

## tStatCatcherStatistics

Select this check box to collect log data at the component level.

## Usage

### Usage rule

This component is mainly used when no particular transformation is required on the data to be loaded onto the database.

### Limitation

The database server must be installed on the same machine where the Studio is installed or where the Job using `tGreenplumOutputBulkExec` is deployed, so that the component functions properly.
Related scenarios

For use cases in relation with `tGreenplumOutputBulkExec`, see the following scenarios:

- Scenario: Inserting transformed data in MySQL database on page 2537.
- Scenario: Inserting data in bulk in MySQL database on page 2544.
**tGreenplumRollback**

Avoids to commit part of a transaction involuntarily.
*tGreenplumRollback* cancels the transaction committed in the connected DB.

**tGreenplumRollback Standard properties**

These properties are used to configure *tGreenplumRollback* running in the Standard Job framework. The *Standard tGreenplumRollback* component belongs to the Databases family. The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tGreenplumConnection</strong> component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other *tGreenplum* components, especially with the **tGreenplumConnection** and **tGreenplumCommit** components. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |
Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

## Related scenarios

For **tGreenplumRollback** related scenario, see **Scenario: Rollback from inserting data in mother/daughter tables** on page 2480.
tGreenplumRow

Acts on the actual DB structure or on the data (although without handling data), depending on the nature of the query and the database.

The SQLBuilder tool helps you write easily your SQL statements.

tGreenplumRow is the specific component for this database query. It executes the SQL query stated onto the specified database. The row suffix means the component implements a flow in the job design although it doesn’t provide output.

**tGreenplumRow Standard properties**

These properties are used to configure tGreenplumRow running in the Standard Job framework.

The Standard tGreenplumRow component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
</tbody>
</table>
2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. **Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.
| **Repository** | You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*. |
| **Click Edit schema** | to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
<p>| <strong>Table Name</strong> | Name of the table to be read. |
| <strong>Query type</strong> | Either Built-in or Repository. |</p>
<table>
<thead>
<tr>
<th><strong>Built-in</strong>: Fill in manually the query statement or build it graphically using SQLBuilder.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Repository</strong>: Select the relevant query stored in the Repository. The <strong>Query</strong> field gets accordingly filled in.</td>
</tr>
<tr>
<td><strong>Guess Query</strong>: Click the <strong>Guess Query</strong> button to generate the query which corresponds to your table schema in the <strong>Query</strong> field.</td>
</tr>
<tr>
<td><strong>Query</strong>: Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Die on error</strong>: This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Propagate QUERY's recordset</strong>: Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the <strong>use column</strong> list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td>This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY's recordset should be set to the type of <strong>Object</strong> and this component is usually followed by <strong>tParseRecordSet</strong>.</td>
</tr>
<tr>
<td><strong>Use PreparedStatement</strong>: Select this check box if you want to query the database using a PreparedStatement. In the <strong>Set PreparedStatement Parameter</strong> table, define the represented by &quot;?&quot; in the SQL instruction of the <strong>Query</strong> field in the <strong>Basic Settings</strong> tab.</td>
</tr>
<tr>
<td><strong>Parameter Index</strong>: Enter the parameter position in the SQL instruction.</td>
</tr>
<tr>
<td><strong>Parameter Type</strong>: Enter the parameter type.</td>
</tr>
<tr>
<td><strong>Parameter Value</strong>: Enter the parameter value.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td>This option is very useful if you need to execute the same query several times. Performance levels are increased</td>
</tr>
<tr>
<td><strong>Commit every</strong>: Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong>: Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
Global Variables

| Global Variables | QUERY: the query statement being processed. This is a Flow variable and it returns a string.  
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component offers the flexibility of the DB query and covers all possible SQL queries. |

Dynamic settings

| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  
The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.  
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

Related scenarios

For a related scenario, see:

- Scenario: Combining two flows for selective output on page 2558
- Procedure on page 625.
• **Scenario: Removing and regenerating a MySQL table index** on page 2552.
**tGreenplumSCD**

Addresses Slowly Changing Dimension needs, reading regularly a source of data and logging the changes into a dedicated SCD table.

tGreenplumSCD reflects and tracks changes in a dedicated Greenplum SCD table.

**tGreenplumSCD Standard properties**

These properties are used to configure tGreenplumSCD running in the Standard Job framework.

The Standard tGreenplumSCD component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

---

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

---

### Basic settings

<table>
<thead>
<tr>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
</tr>
<tr>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
</tr>
<tr>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
</tr>
<tr>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
</tr>
<tr>
<td>Select the Repository file where properties are stored. The following fields are pre-filled in using fetched data.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
</tr>
<tr>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

---

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide.*

<table>
<thead>
<tr>
<th><strong>Connection type</strong></th>
<th>Select the relevant driver on the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Name of the DB schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>The schema is created and stored locally for this component only. Related topic: see <em>Talend Studio User Guide.</em></td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <em>Talend Studio User Guide.</em></td>
</tr>
<tr>
<td><strong>SCD Editor</strong></td>
<td>The SCD editor helps to build and configure the data flow for slowly changing dimension outputs. For more information, see SCD management methodology on page 2566.</td>
</tr>
</tbody>
</table>
### Use memory saving Mode

Select this check box to maximize system performance.

### Source keys include Null

Select this check box to allow the source key columns to have Null values.

**Warning:**
Special attention should be paid to the uniqueness of the source key(s) value when this option is selected.

### Die on error

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

### Advanced settings

**End date time details**

Specify the time value of the SCD end date time setting in the format of `HH:mm:ss`. The default value for this field is `12:00:00`. This field appears only when SCD Type 2 is used and Fixed year value is selected for creating the SCD end date.

**Debug mode**

Select this check box to display each step during processing entries in a database.

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

### Global Variables

**Global Variables**

- **NB_LINE_UPDATED**: the number of rows updated. This is an After variable and it returns an integer.
- **NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.
- **NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).
### Usage

<table>
<thead>
<tr>
<th><strong>Usage rule</strong></th>
<th>This component is used as Output component. It requires an Input component and Row main link as input.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamic settings</strong></td>
<td>Click the [*] button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The <strong>Dynamic settings</strong> table is available only when the <strong>Use an existing connection</strong> check box is selected in the <strong>Basic settings</strong> view. Once a dynamic parameter is defined, the <strong>Component List</strong> box in the <strong>Basic settings</strong> view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on <strong>Dynamic settings</strong> and context variables, see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Limitation</strong></td>
<td>This component does not support using SCD type 0 together with other SCD types.</td>
</tr>
</tbody>
</table>

### Related scenario

For related scenarios, see **tMysqlSCD** on page 2563.
**tGroovy**

tGroovy broadens the functionality of the **Talend** Job, using the Groovy language which is a simplified Java syntax.

tGroovy allows you to enter customized code which you can integrate in the **Talend** programme. The code is run only once.

**tGroovy Standard properties**

These properties are used to configure tGroovy running in the Standard Job framework.

The Standard tGroovy component belongs to the Custom Code family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Groovy Script</th>
<th>Enter the Groovy code you want to run.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>This table has two columns.</td>
</tr>
<tr>
<td></td>
<td><strong>Name</strong>: Name of the variable called in the code.</td>
</tr>
<tr>
<td></td>
<td><strong>Value</strong>: Value associated with the variable.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect the log data at component level.</th>
</tr>
</thead>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used alone or as a subjob along with one other component.</th>
</tr>
</thead>
</table>
Limitation

Knowledge of the Groovy language is required.

**Related Scenarios**

- For a scenario using the Groovy code, see *Scenario: Calling a file which contains Groovy code* on page 1365.
- For a functional example, see *Scenario: Printing out a variable content* on page 1835
tGroovyFile

Broadens the functionality of *Talend* Jobs using the Groovy language which is a simplified Java syntax.

tGroovyFile allows you to call an existing Groovy script.

**tGroovyFile Standard properties**

These properties are used to configure tGroovyFile running in the Standard Job framework.

The Standard tGroovyFile component belongs to the Custom Code family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Groovy File</th>
<th>Name and path of the file containing the Groovy code.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Variables</th>
<th>This table contains two columns.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Name: Name of the variable called in the code.</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>Value: Value associated with this variable.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect the log data at component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the *Die on error* check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press *Ctrl + Space* to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used alone or as a sub-job along with another component.</th>
</tr>
</thead>
</table>

| Limitation | Knowledge of the Groovy language is required. |
Scenario: Calling a file which contains Groovy code

This scenario uses tGroovyFile, on its own. The Job calls a file containing Groovy code in order to display the file information in the Console.

Setting up the Job

Open the Custom_Code folder in the Palette and drop a tGroovyFile component onto the workspace.

Configuring the tGroovyFile component

Procedure

1. Double-click the component to display the Component view.

2. In the Groovy File field, enter the path to the file containing the Groovy code, or browse to the file in your directory. In this example, it is D:/Input/Ageducapitaine.txt, and the file contains the following Groovy codes:

   ```groovy
   println("The captain is "+age+" years old")
   ```

3. In the Variables table, add a line by clicking the [+ ] button.

4. In the Name column, enter "age", and then in the Value column, enter 50.

Executing the Job

Procedure

1. Press Ctrl+S to save your Job.

2. Press F6 or click Run on the Run tab to execute the Job.
The **Console** displays the information contained in the input file, to which the variable result is added.

```
Starting job tGroovyFile at 09:49 17/02/2010.
[statistics] connecting to socket on port 4016
[statistics] connected
The captain is 50 years old
[statistics] disconnected
Job tGroovyFile ended at 09:49 17/02/2010. [exit code=0]
```
tGSBucketCreate

Creates a new bucket which you can use to organize data and control access to data in Google Cloud Storage.

**tGSBucketCreate Standard properties**

These properties are used to configure tGSBucketCreate running in the Standard Job framework.

The Standard tGSBucketCreate component belongs to the Big Data and the Cloud families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Key and Secret Key</strong></td>
<td>Type in the authentication information obtained from Google for making requests to Google Cloud Storage. These keys can be consulted on the Interoperable Access tab view under the Google Cloud Storage tab of the project from the Google APIs Console. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. For more information about the access key and secret key, go to <a href="https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/">https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/</a> and see the description about developer keys. The Access Key and Secret Key fields will be available only if you do not select the Use an existing connection check box.</td>
</tr>
<tr>
<td><strong>Bucket name</strong></td>
<td>Specify the name of the bucket which you want to create. Note that the bucket name must be unique across the Google Cloud Storage system. For more information about the bucket naming convention, see <a href="https://developers.google.com/storage/docs/bucketnaming">https://developers.google.com/storage/docs/bucketnaming</a>.</td>
</tr>
<tr>
<td><strong>Special configure</strong></td>
<td>Select this check box to provide the additional configuration for the bucket to be created.</td>
</tr>
<tr>
<td><strong>Project ID</strong></td>
<td>Specify the project ID to which the new bucket belongs.</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>Select from the list the location where the new bucket will be created. Currently, Europe and US are available. By default, the bucket location is in the US.</td>
</tr>
</tbody>
</table>
Note that once a bucket is created in a specific location, it cannot be moved to another location.

**Acl**

Select from the list the desired access control list (ACL) for the new bucket.

Depending on the ACL on the bucket, the access requests from users may be allowed or rejected. If you do not specify a predefined ACL for the new bucket, the predefined project-private ACL applies.

For more information about ACL, see https://developers.google.com/storage/docs/accesscontrol?hl=en.

**Die on error**

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

**Advanced settings**

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.

**Usage**

**Usage rule**

This component can be used together with the **tGSBucketList** component to check if a new bucket is created successfully.

**Related scenario**

For related topics, see **Scenario: Verifying the absence of a bucket, creating it and listing all the S3 buckets** on page 3324.
tGSBucketDelete

Deletes an empty bucket in Google Cloud Storage so as to release occupied resources.

Note that bucket deletion cannot be undone, so you need to back up any data that you want to keep before the deletion.

**tGSBucketDelete Standard properties**

These properties are used to configure tGSBucketDelete running in the Standard Job framework.

The Standard tGSBucketDelete component belongs to the Big Data and the Cloud families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Key and Secret Key</td>
<td>Type in the authentication information obtained from Google for making requests to Google Cloud Storage. These keys can be consulted on the Interoperable Access tab view under the Google Cloud Storage tab of the project from the Google APIs Console. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. For more information about the access key and secret key, go to <a href="https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/">https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/</a> and see the description about developer keys. The Access Key and Secret Key fields will be available only if you do not select the Use an existing connection check box.</td>
</tr>
<tr>
<td>Bucket name</td>
<td>Specify the name of the bucket that you want to delete. Make sure that the bucket to be deleted is empty.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics     | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |
### Global Variables

| Global Variables |  
|------------------|------------------|
| **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see [Talend Studio User Guide](#). |

### Usage

| Usage rule | This component can be used together with the **tGSBucketList** component to check if the specified bucket is deleted successfully. |

### Related scenarios

No scenario is available for the Standard version of this component yet.
tGSBucketExist

Checks the existence of a bucket in Google Cloud Storage so as to make further operations.

**tGSBucketExist Standard properties**

These properties are used to configure tGSBucketExist running in the Standard Job framework.

The Standard tGSBucketExist component belongs to the Big Data and the Cloud families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Use an existing connection</strong></th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Key and Secret Key</strong></td>
<td>Type in the authentication information obtained from Google for making requests to Google Cloud Storage. These keys can be consulted on the Interoperable Access tab view under the Google Cloud Storage tab of the project from the Google APIs Console. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. For more information about the access key and secret key, go to <a href="https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/">https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/</a> and see the description about developer keys. The Access Key and Secret Key fields will be available only if you do not select the Use an existing connection check box.</td>
</tr>
<tr>
<td><strong>Bucket name</strong></td>
<td>Specify the name of the bucket for which you want to perform a check to confirm it exists in Google Cloud Storage.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level.                                                                                       |
Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>BUCKET_EXIST: the existence of a specified bucket. This is a Flow variable and it returns a boolean.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BUCKET_NAME: the name of a specified bucket. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule | This component can be used as a standalone component. |

Related scenario

For related topics, see Scenario: Verifying the absence of a bucket, creating it and listing all the S3 buckets on page 3324.
**tGSBucketList**

Retrieves a list of buckets from all projects or one specific project in Google Cloud Storage.

*tGSBucketList* iterates on all buckets within all projects or one specific project in Google Cloud Storage.

**tGSBucketList Standard properties**

These properties are used to configure *tGSBucketList* running in the Standard Job framework.

The Standard *tGSBucketList* component belongs to the Big Data and the Cloud families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Key</strong> and <strong>Secret Key</strong></td>
<td>Type in the authentication information obtained from Google for making requests to Google Cloud Storage. These keys can be consulted on the Interoperable Access tab view under the Google Cloud Storage tab of the project from the Google APIs Console. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings. For more information about the access key and secret key, go to <a href="https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/">https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/</a> and see the description about developer keys. The Access Key and Secret Key fields will be available only if you do not select the <strong>Use an existing connection</strong> check box.</td>
</tr>
<tr>
<td><strong>Specify project ID</strong></td>
<td>Select this check box and in the <strong>Project ID</strong> field specify a project ID from which you want to retrieve a list of buckets.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

**Global Variables**

| Global Variables | **CURRENT_BUCKET_NAME**: the current bucket name. This is a Flow variable and it returns a string. |
**tGSBucketList**

**NB_BUCKET**: the number of buckets. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>The <strong>tGSBucketList</strong> component can be used as a standalone component or as a start component of a proc ess.</th>
</tr>
</thead>
</table>

## Related scenario

For related topics, see *Scenario: Verifying the absence of a bucket, creating it and listing all the S3 buckets* on page 3324.
tGSClose

Closes an active connection to Google Cloud Storage in order to release the occupied resources.

**tGSClose Standard properties**

These properties are used to configure tGSClose running in the Standard Job framework.

The Standard tGSClose component belongs to the Big Data and the Cloud families.

The component in this framework is generally available.

**Basic settings**

| Component List | Select the *tGSConnection* component in the list if more than one connection is planned for the current Job. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the *Die on error* check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press *Ctrl + Space* to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

**Usage**

| Usage rule | This component is generally used with other Google Cloud Storage components, particularly *tGSConnection*. |

**Related scenario**

For a scenario in which *tGSClose* is used, see *Scenario: Managing files with Google Cloud Storage* on page 1389.
tGSCollection

Provides the authentication information for making requests to the Google Cloud Storage system and enables the reuse of the connection it creates to Google Cloud Storage.

**tGSCollection Standard properties**

These properties are used to configure tGSCollection running in the Standard Job framework. The Standard tGSCollection component belongs to the Big Data and the Cloud families. The component in this framework is generally available.

**Basic settings**

| Access Key and Secret Key | Type in the authentication information obtained from Google for making requests to Google Cloud Storage. These keys can be consulted on the Interoperable Access tab view under the Google Cloud Storage tab of the project from the Google APIs Console. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. 
For more information about the access key and secret key, go to [https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en](https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en) and see the description about developer keys. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see **Talend Studio User Guide.** |

Usage

| Usage rule | This component is generally used with other Google Cloud Storage components, particularly **tGSClose**. |

**Related scenario**

For a scenario in which **tGSCconnection** is used, see **Scenario: Managing files with Google Cloud Storage** on page 1389.
**tGSCopy**

Copies or moves objects within a bucket or between buckets in Google Cloud Storage.
tGSCopy streamlines processes by automating the copy tasks.

**tGSCopy Standard properties**

These properties are used to configure tGSCopy running in the Standard Job framework.
The Standard tGSCopy component belongs to the Big Data and the Cloud families.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Access Key</strong> and <strong>Secret Key</strong></td>
<td>Type in the authentication information obtained from Google for making requests to Google Cloud Storage. These keys can be consulted on the Interoperable Access tab view under the Google Cloud Storage tab of the project from the Google APIs Console. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. For more information about the access key and secret key, go to <a href="https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/">https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/</a> and see the description about developer keys. The Access Key and Secret Key fields will be available only if you do not select the Use an existing connection check box.</td>
</tr>
<tr>
<td>Source bucket name</td>
<td>Specify the name of the bucket from which you want to copy or move objects.</td>
</tr>
<tr>
<td>Source object key</td>
<td>Specify the key of the object to be copied.</td>
</tr>
<tr>
<td>Source is folder</td>
<td>Select this check box if the source object is a folder.</td>
</tr>
<tr>
<td>Target bucket name</td>
<td>Specify the name of the bucket to which you want to copy or move objects.</td>
</tr>
<tr>
<td>Target folder</td>
<td>Specify the target folder to which the objects will be copied or moved.</td>
</tr>
<tr>
<td><strong>Action</strong></td>
<td>Select the action that you want to perform on objects from the list.</td>
</tr>
<tr>
<td>• Copy</td>
<td>copies objects from the source bucket or folder to the target bucket or folder.</td>
</tr>
</tbody>
</table>
- **Move**: moves objects from the source bucket or folder to the target bucket or folder.

### Rename

Select this check box and in the **New name** field enter a new name for the object to be copied or moved.

The **Rename** check box will not be available if you select the **Source is folder** check box.

### Die on error

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**Global Variables**

- **SOURCE_BUCKET**: the source bucket name. This is an After variable and it returns a string.
- **SOURCE_OBJECTKEY**: the key of a source object. This is an After variable and it returns a string.
- **DESTINATION_BUCKETNAME**: the destination bucket name. This is an After variable and it returns a string.
- **DESTINATION_FOLDER**: the destination folder. This is an After variable and it returns a string.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

**Usage rule**

This component can be used as a standalone component.
Related scenario

For a scenario in which tGSCopy is used, see Scenario: Managing files with Google Cloud Storage on page 1389.
tGSDelete

Deletes the objects which match the specified criteria in Google Cloud Storage so as to release the occupied resources.

tGSDelete Standard properties

These properties are used to configure tGSDelete running in the Standard Job framework.

The Standard tGSDelete component belongs to the Big Data and the Cloud families.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Key and Secret Key</strong></td>
<td>Type in the authentication information obtained from Google for making requests to Google Cloud Storage. These keys can be consulted on the Interoperable Access tab view under the Google Cloud Storage tab of the project from the Google APIs Console. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. For more information about the access key and secret key, go to <a href="https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/">https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/</a> and see the description about developer keys. The Access Key and Secret Key fields will be available only if you do not select the Use an existing connection check box.</td>
</tr>
<tr>
<td>Key prefix</td>
<td>Specify the prefix to delete only objects whose keys begin with the specified prefix.</td>
</tr>
<tr>
<td>Delimiter</td>
<td>Specify the delimiter in order to delete only those objects with key names up to the delimiter.</td>
</tr>
<tr>
<td>Specify project ID</td>
<td>Select this check box and in the Project ID field enter the project ID from which you want to delete objects.</td>
</tr>
<tr>
<td>Delete object from bucket list</td>
<td>Select this check box and complete the Bucket table to delete objects in the specified buckets. • <strong>Bucket name</strong>: type in the name of the bucket from which you want to delete objects. • <strong>Key prefix</strong>: type in the prefix to delete objects whose keys begin with the specified prefix in the specified bucket.</td>
</tr>
</tbody>
</table>
**tGSDelete**

- **Delimiter**: type in the delimiter to delete those objects with key names up to the delimiter in the specified bucket.

  If you select the **Delete object from bucket list** check box, the **Key prefix** and **Delimiter** fields as well as the **Specify project ID** check box will not be available.

<table>
<thead>
<tr>
<th><strong>Die on error</strong></th>
<th>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</th>
</tr>
</thead>
</table>

**Advanced settings**

- **tStatCatcher Statistics**
  
  Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

- **Global Variables**
  
  **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

  **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

  A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

  To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

  For further information about variables, see **Talend Studio User Guide**.

**Usage**

- **Usage rule**
  
  This component can be used together with the **tGSList** component to check if the objects which match the specified criteria are deleted successfully.

**Related scenario**

For a scenario in which **tGSDelete** is used, see **Scenario: Managing files with Google Cloud Storage** on page 1389.
tGSGet

Retrieves objects which match the specified criteria from Google Cloud Storage and outputs them to a local directory.

**tGSGet Standard properties**

These properties are used to configure tGSGet running in the Standard Job framework.

The Standard tGSGet component belongs to the Big Data and the Cloud families.

The component in this framework is generally available.

**Basic settings**

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Access Key and Secret Key**

Type in the authentication information obtained from Google for making requests to Google Cloud Storage.

These keys can be consulted on the Interoperable Access tab view under the Google Cloud Storage tab of the project from the Google APIs Console.

To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

For more information about the access key and secret key, go to [https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/](https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/) and see the description about developer keys.

The **Access Key** and **Secret Key** fields will be available only if you do not select the **Use an existing connection** check box.

**Key prefix**

Specify the prefix to download only objects which keys begin with the specified prefix.

**Delimiter**

Specify the delimiter in order to download only those objects with key names up to the delimiter.

**Specify project ID**

Select this check box and in the **Project ID** field enter the project ID from which you want to obtain objects.

**Use keys**

Select this check box and complete the **Keys** table to define the criteria for objects to be downloaded from Google Cloud Storage.

- **Bucket name**: type in the name of the bucket from which you want to download objects.
- **Key**: type in the key of the object to be downloaded.
- **New name**: type in a new name for the object to be downloaded.

If you select the **Use keys** check box, the **Key prefix** and **Delimiter** fields as well as the **Specify project ID** check box and the **Get files from bucket list** check box will not be available.

### Get files from bucket list

Select this check box and complete the **Bucket** table to define the criteria for objects to be downloaded from Google Cloud Storage.

- **Bucket name**: type in the name of the bucket from which you want to download objects.
- **Key prefix**: type in the prefix to download objects whose keys start with the specified prefix from the specified bucket.
- **Delimiter**: specify the delimiter to download those objects with key names up to the delimiter from the specified bucket.

If you select the **Get files from bucket list** check box, the **Key prefix** and **Delimiter** fields as well as the **Specify project ID** check box and the **Use keys** check box will not be available.

### Output directory

Specify the directory where you want to store the downloaded objects.

### Die on error

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

### Advanced settings

- **tStatCatcher Statistics**
  
  Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

- **Global Variables**

  - **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

  - **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

  A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

  To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
For further information about variables, see *Talend Studio User Guide*.

## Usage

| Usage rule | This component is usually used together with other Google Cloud Storage components, particularly *tGSPut*. |

## Related scenarios

No scenario is available for the Standard version of this component yet.
tGSLList

Retrieves a list of objects from Google Cloud Storage one by one.
tGSLList iterates on a list of objects which match the specified criteria in Google Cloud Storage.

**tGSLList Standard properties**

These properties are used to configure tGSLList running in the Standard Job framework.
The Standard tGSLList component belongs to the Big Data and the Cloud families.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Key and Secret Key</td>
<td>Type in the authentication information obtained from Google for making requests to Google Cloud Storage. These keys can be consulted on the Interoperable Access tab view under the Google Cloud Storage tab of the project from the Google APIs Console. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. For more information about the access key and secret key, go to <a href="https://developers.google.com/storage/docs/reference/v1/getting-started?hl=en/">https://developers.google.com/storage/docs/reference/v1/getting-started?hl=en/</a> and see the description about developer keys. The Access Key and Secret Key fields will be available only if you do not select the Use an existing connection check box.</td>
</tr>
<tr>
<td>Key prefix</td>
<td>Specify the key prefix so that only the objects whose keys begin with the specified string will be listed.</td>
</tr>
<tr>
<td>Delimiter</td>
<td>Specify the delimiter in order to list only those objects with key names up to the delimiter.</td>
</tr>
<tr>
<td>Specify project ID</td>
<td>Select this check box and in the Project ID field enter the project ID from which you want to retrieve a list of objects.</td>
</tr>
</tbody>
</table>
| List objects in bucket list | Select this check box and complete the Bucket table to retrieve objects in the specified buckets.  
- **Bucket name**: type in the name of the bucket from which you want to retrieve objects. |
Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Global Variables

**Global Variables**

- **CURRENT_BUCKET**: the current bucket name. This is a Flow variable and it returns a string.
- **CURRENT_KEY**: the current key. This is a Flow variable and it returns a string.
- **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**

The **tGSList** component can be used as a standalone component or as a start component of a process.

Related scenario

For a scenario in which **tGSList** is used, see *Scenario: Managing files with Google Cloud Storage* on page 1389.
**tGSPut**

Uploads files from a local directory to Google Cloud Storage so that you can manage them with Google Cloud Storage.

**tGSPut Standard properties**

These properties are used to configure tGSPut running in the Standard Job framework.

The Standard tGSPut component belongs to the Big Data and the Cloud families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Key and Secret Key</strong></td>
<td>Type in the authentication information obtained from Google for making requests to Google Cloud Storage. These keys can be consulted on the Interoperable Access tab view under the Google Cloud Storage tab of the project from the Google APIs Console. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. For more information about the access key and secret key, go to <a href="https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/">https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/</a> and see the description about developer keys. The <strong>Access Key</strong> and <strong>Secret Key</strong> fields will be available only if you do not select the <strong>Use an existing connection</strong> check box.</td>
</tr>
<tr>
<td><strong>Bucket name</strong></td>
<td>Type in the name of the bucket into which you want to upload files.</td>
</tr>
<tr>
<td><strong>Local directory</strong></td>
<td>Type in the full path of or browse to the local directory where the files to be uploaded are located.</td>
</tr>
<tr>
<td><strong>Google Storage directory</strong></td>
<td>Type in the Google Storage directory to which you want to upload files.</td>
</tr>
</tbody>
</table>
| **Use files list** | Select this check box and complete the **Files** table.  
- **Filemask**: enter the filename or filemask using wildcharacters (*) or regular expressions.  
- **New name**: enter a new name for the file after being uploaded. |
Die on error

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Global Variables

**Global Variables**

- **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**

This component can be used together with other components, particularly the **tGSGet** component.

Scenario: Managing files with Google Cloud Storage

The scenario describes a Job which uploads files from the local directory to a bucket in Google Cloud Storage, then performs copy, move and delete operations on those files, and finally lists and displays the files in relevant buckets on the console.
**Prerequisites:** You have purchased a Google Cloud Storage account and created three buckets under the same Google Storage directory. In this example, the buckets created are `bighouse`, `bed_room`, and `study_room`.

### Dropping and linking the components

**About this task**

To design the Job, proceed as follows:

**Procedure**

1. Drop the following components from the **Palatte** to design the workspace: one `tGSCconnection` component, one `tGSPut` component, two `tGSCopy` components, one `tGSDelete` component, one
tGSPut component, one tIterateToFlow component, one tLogRow component and one tGSClose component.

2. Connect tGSConnection to tGSPut using a Trigger > On Subjob Ok link.
3. Connect tGSPut to the first tGSCopy using a Trigger > On Subjob Ok link.
4. Do the same to connect the first tGSCopy to the second tGSCopy, connect the second tGSCopy to tGSDelete, connect tGSDelete to tGSList, and connect tGSList to tGSClose.
5. Connect tGSList to tIterateToFlow using a Row > Iterate link.
6. Connect tIterateToFlow to tLogRow using a Row > Main link.

Configuring the components

Opening a connection to Google Cloud Storage

Procedure

1. Double-click the tGSConnection component to open its Basic settings view in the Component tab.

2. Navigate to the Google APIs Console in your web browser to access the Google project hosting the Cloud Storage services you need to use.
3. Click Google Cloud Storage > Interoperable Access to open its view, and copy the access key and secret key.
4. In the Component view of the Studio, paste the access key and secret key to the corresponding fields respectively.

Uploading files to Google Cloud Storage

Procedure

1. Double-click the tGSPut component to open its Basic settings view in the Component tab.

2. Select the Use an existing connection check box and then select the connection you have configured earlier.
3. In the Bucket name field, enter the name of the bucket into which you want to upload files. In this example, bighouse.
4. In the Local directory field, browse to the directory from which the files will be uploaded, D:/Input/House in this example.
The files under this directory are shown below:

```
Name
books.txt
clothes.txt
computer_01.txt
computer_02.txt
computer_03.csv
model.csv
bigs.txt
tables.txt
Toys.txt
```

5. Leave other settings as they are.

**Copying all files from one bucket to another bucket**

**Procedure**

1. Double-click the first \texttt{tGSCopy} component to open its \texttt{Basic settings} view in the \texttt{Component} tab.

   ![Component Settings](image)

2. Select the \texttt{Use an existing connection} check box and then select the connection you have configured earlier.

3. In the \texttt{Source bucket name} field, enter the name of the bucket from which you want to copy files, \texttt{bighouse} in this example.

4. Select the \texttt{Source is a folder} check box. All files from the bucket \texttt{bighouse} will be copied.

5. In the \texttt{Target bucket name} field, enter the name of the bucket into which you want to copy files, \texttt{bed_room} in this example.

6. Select \texttt{Copy} from the \texttt{Action} list.

**Moving a file from one bucket to another bucket and renaming it**

**Procedure**

1. Double-click the second \texttt{tGSCopy} component to open its \texttt{Basic settings} view in the \texttt{Component} tab.

   ![Component Settings](image)
2. Select the **Use an existing connection** check box and then select the connection you have configured earlier.

3. In the **Source bucket name** field, enter the name of the bucket from which you want to move files, *bighouse* in this example.

4. In the **Source object key** field, enter the key of the object to be moved, *computer_01.txt* in this example.

5. In the **Target bucket name** field, enter the name of the bucket into which you want to move files, *study_room* in this example.

6. Select **Move** from the **Action** list. The specified source file *computer_01.txt* will be moved from the bucket *bighouse* to *study_room*.

7. Select the **Rename** check box. In the **New name** field, enter a new name for the moved file. In this example, the new name is *laptop.txt*.

8. Leave other settings as they are.

---

### Deleting a file in one bucket

**Procedure**

1. Double-click the **tGSDel**elete component to open its **Basic settings** view in the **Component** tab.

   ![Image of tGSDel component](image)

   - **Connection**: Use an existing connection
   - **Delete configuration**: Delete object from bucket list

   **Bucket**
   - **Bucket name**: "bed_room"
   - **Key prefix**: "computer_03"

2. Select the **Use an existing connection** check box and then select the connection you have configured earlier.
3. Select the **Delete object from bucket list** check box. Fill in the **Bucket** table with the file information that you want to delete.

   In this example, the file *computer_03.csv* will be deleted from the bucket *bed_room* whose files are copied from the bucket *bighouse*.

**Listing all files in the three buckets**

**Procedure**

1. Double-click the **tGSList** component to open its **Basic settings** view in the **Component** tab.

   ![tGSList component settings](image)

   1. Select the **Use an existing connection** check box and then select the connection you have configured earlier.
   2. Select the **List objects in bucket list** check box. In the **Bucket** table, enter the name of the three buckets in the **Bucket name** column, *bighouse*, *study_room*, and *bed_room*.
   3. Double-click the **tIterateToFlow** component to open its **Basic settings** view in the **Component** tab.

   ![tIterateToFlow component settings](image)

   5. Click **Edit schema** to define the data to pass on to **tLogRow**.

   In this example, add two columns *bucketName* and *key*, and set their types to **Object**.
6. The **Mapping** table will be populated with the defined columns automatically.
   In the **Value** column, enter `globalMap.get("tGSList_2_CURRENT_BUCKET")` for the **bucketName** column and `globalMap.get("tGSList_2_CURRENT_KEY")` for the **key** column. You can also press **Ctrl + Space** and then choose the appropriate variable.

7. Double-click the **tLogRow** component to open its **Basic settings** view in the **Component** tab.

8. Select **Table (print values in cells of a table)** for a better view of the results.

**Closing the connection to Google Cloud Storage**

**Procedure**

1. Double-click the **tGSClose** component to open its **Basic settings** view in the **Component** tab.
2. Select the connection you want to close from the **Component List**.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Execute the Job by pressing **F6** or clicking **Run** on the **Run** tab.
The files in the three buckets are displayed. As expected, at first, the files from the bucket bighouse are copied to the bucket bed_room, then the file computer_01.txt from the bucket bighouse is moved to the bucket study_room and renamed to be laptop.txt, finally the file computer_03.csv is deleted from the bucket bed_room.

<table>
<thead>
<tr>
<th>tLogRow_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucketName</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>bighouse</td>
</tr>
<tr>
<td>bighouse</td>
</tr>
<tr>
<td>bighouse</td>
</tr>
<tr>
<td>bighouse</td>
</tr>
<tr>
<td>bighouse</td>
</tr>
<tr>
<td>bighouse</td>
</tr>
<tr>
<td>bighouse</td>
</tr>
<tr>
<td>study_room</td>
</tr>
<tr>
<td>bed_room</td>
</tr>
<tr>
<td>bed_room</td>
</tr>
<tr>
<td>bed_room</td>
</tr>
<tr>
<td>bed_room</td>
</tr>
<tr>
<td>bed_room</td>
</tr>
<tr>
<td>bed_room</td>
</tr>
<tr>
<td>bed_room</td>
</tr>
</tbody>
</table>
tHashInput

Reads from the cache memory data loaded by tHashOutput to offer high-speed data feed, facilitating transactions involving a large amount of data.

The components of the Technical family are normally hidden from the Palette by default. For more information about how to show them on the Palette, see Talend Studio User Guide.

tHashInput Standard properties

These properties are used to configure tHashInput running in the Standard Job framework.

The Standard tHashInput component belongs to the Technical family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either built-in or remotely stored in the Repository. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td>Built-in: The schema is created and stored locally for this component only. Related topic: see the Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Repository: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see the Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Link with a tHashOutput</td>
<td>Select this check box to connect to a tHashOutput component. It is always selected by default.</td>
</tr>
<tr>
<td>Component list</td>
<td>Drop-down list of available tHashOutput components.</td>
</tr>
<tr>
<td>Clear cache after reading</td>
<td>Select this check box to clear the cache after reading the data loaded by a certain tHashOutput component. This way, the following tHashInput components, if any,</td>
</tr>
</tbody>
</table>
Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

Global Variables

| Global Variables | tHashInput | will not be able to read the cached data loaded by that tHashOutput component. |

Usage

| Usage rule | tHashInput | This component is used along with tHashOutput. It reads from the cache memory data loaded by tHashOutput. Together, these twin components offer high-speed data access to facilitate transactions involving a massive amount of data. |

Scenario 1: Reading data from the cache memory for high-speed data access

The following Job reads from the cache memory a huge amount of data loaded by two tHashOutput components and pass it to a tFileOutputDelimited. The goal of this scenario is to show the speed at which mass data is read and written. In practice, data feed generated in this way can be used as lookup table input for some use cases where a big amount of data needs to be referenced.

Dropping and linking the components

Procedure

1. Drag and drop the following components from the Palette to the workspace: tFixedFlowInput (X2), tHashOutput (X2), tHashInput and tFileOutputDelimited.
2. Connect the first tFixedFlowInput to the first tHashOutput using a Row > Main link.
3. Connect the second tFixedFlowInput to the second tHashOutput using a Row > Main link.
4. Connect the first subjob (from `tFixedFlowInput_1`) to the second subjob (to `tFixedFlowInput_2`) using an **OnSubjobOk** link.

5. Connect `tHashInput` to `tFileOutputDelimited` using a **Row > Main** link.

6. Connect the second subjob to the last subjob using an **OnSubjobOk** link.

---

**Configuring the components**

**Configuring data inputs and hash cache**

**Procedure**

1. Double-click the first `tFixedFlowInput` component to display its **Basic settings** view.

   ![Diagram](image)

   **Basic settings**
   - **Schema**: Built-In
   - **Number of rows**: 50000
   - **Mode**: Use Single Table
   - **Values**:
     - ID
     - ID_Insurance
   - **Use Inline Table**
   - **Use Inline Content (delimited file)**

2. Select **Built-In** from the **Schema** drop-down list.

   **Note:**
You can select **Repository** from the **Schema** drop-down list to fill in the relevant fields automatically if the relevant metadata has been stored in the **Repository**. For more information about **Metadata**, see the **Talend Studio User Guide**.

3. Click **Edit schema** to define the data structure of the input flow. In this case, the input has two columns: **ID** and **ID_Insurance**, and then click **OK** to close the dialog box.

4. Fill in the **Number of rows** field to specify the entries to output, e.g. **50000**.

5. Select the **Use Single Table** check box. In the **Values** table and in the **Value** column, assign values to the columns, e.g. 1 for **ID** and 3 for **ID_Insurance**.

6. Perform the same operations for the second **tFixedFlowInput** component, with the only difference in the values. That is, 2 for **ID** and 4 for **ID_Insurance** in this case.

7. Double-click the first **tHashOutput** to display its **Basic settings** view.

8. Select **Built-In** from the **Schema** drop-down list and click **Sync columns** to retrieve the schema from the previous component. Select **Keep all** from the **Keys management** drop-down list and keep the **Append** check box selected.

9. Perform the same operations for the second **tHashOutput** component, and select the **Link with a tHashOutput** check box.

**Configuring data retrieval from hash cache and data output**

**Procedure**

1. Double-click **tHashInput** to display its **Basic settings** view.
2. Select **Built-In** from the **Schema** drop-down list. Click **Edit schema** to define the data structure, which is the same as that of **tHashOutput**.

3. Select **tHashOutput_1** from the **Component list** drop-down list.

4. Double-click **tFileOutputDelimited** to display its **Basic settings** view.

5. Select **Built-In** from the **Property Type** drop-down list. In the **File Name** field, enter the full path and name of the file, e.g. “E:/Allr70207V5.0/Talend-All-r70207-V5.0.0NB/workspace/out.csv”.

6. Select the **Include Header** check box and click **Sync columns** to retrieve the schema from the previous component.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6**, or click **Run** on the **Run** tab to execute the Job.
**Results**

You can find that mass entries are written and read very rapidly.

**Scenario 2: Clearing the memory before loading data to it in case an iterator exists in the same subjob**

In this scenario, the usage of the **Append** option of **tHashOutput** is demonstrated as it helps remove repetitive or unwanted data in case an iterator exists in the same subjob as **tHashOutput**.

To build the Job, do the following:

**Dropping and linking the components**

**Procedure**

1. Drag and drop the following components from the Palette to the workspace: **tLoop**, **tFixedFlowInput**, **tHashOutput**, **tHashInput** and **tLogRow**.
2. Connect **tLoop** to **tFixedFlowInput** using a **Row > Iterate** link.
3. Connect **tFixedFlowInput** to **tHashOutput** using a **Row > Main** link.
4. Connect **tHashInput** to **tLogRow** using a **Row > Main** link.
5. Connect **tLoop** to **tHashInput** using an **OnSubjobOk** link.
Configuring the components

Configuring data input and hash cache

Procedure

1. Double-click the tLoop component to display its Basic settings view.

2. Select For as the loop type. Type in 1, 2, 1 in the From, To and Step fields respectively. Keep the Values are increasing check box selected.

3. Double-click the tFixedFlowInput component to display its Basic settings view.

4. Select Built-In from the Schema drop-down list.

   Note:
   You can select Repository from the Schema drop-down list to fill in the relevant fields automatically if the relevant metadata has been stored in the Repository. For more information about Metadata, see the Talend Studio User Guide.

5. Click Edit schema to define the data structure of the input flow. In this case, the input has one column: Name.
6. Click OK to close the dialog box.

7. Fill in the Number of rows field to specify the entries to output, for example 1.

8. Select the Use Single Table check box. In the Values table, assign a value to the Name field, e.g. Marx.

9. Double-click tHashOutput to display its Basic settings view.

10. Select Built-In from the Schema drop-down list and click Sync columns to retrieve the schema from the previous component. Select Keep all from the Keys management drop-down list and deselect the Append check box.

Configuring data retrieval from hash cache and data output

Procedure

1. Double-click tHashInput to display its Basic settings view.

2. Select Built-In from the Schema drop-down list. Click Edit schema to define the data structure, which is the same as that of tHashOutput.

3. Select tHashOutput_2 from the Component list drop-down list.

4. Double-click tLogRow to display its Basic settings view.
5. Select **Built-In** from the **Schema** drop-down list and click **Sync columns** to retrieve the schema from the previous component. In the **Mode** area, select **Table (print values in cells of a table)**.

### Saving and executing the Job

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6**, or click **Run** on the **Run** tab to execute the Job.

You can find that only one row was output although two rows were generated by **tFixedFlowInput**.
tHashOutput

Loads data to the cache memory to offer high-speed access, facilitating transactions involving a large amount of data.

The components of the Technical family are normally hidden from the Palette by default. For more information about how to show them on the Palette, see Talend Studio User Guide.

**tHashOutput Standard properties**

These properties are used to configure tHashOutput running in the Standard Job framework.

The Standard tHashOutput component belongs to the Technical family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Schema and Edit schema</strong></th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either built-in or remotely stored in the Repository. Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Sync columns</strong> to retrieve the schema from the previous component connected in the Job.</td>
</tr>
</tbody>
</table>

| **Built-in** | The schema is created and stored locally for this component only. Related topic: see the Talend Studio User Guide. |
| **Repository** | The schema already exists and is stored in the Repository, hence can be reused. Related topic: see the Talend Studio User Guide. |

**Link with a tHashOutput**  
Select this check box to connect to a tHashOutput component.

1 **Note:**  
If multiple tHashOutput components are linked in this way, the data loaded to the cache
by all of them can be read by a **tHashInput** component that is linked with any of them.

<table>
<thead>
<tr>
<th>Component list</th>
<th>Drop-down list of available <strong>tHashOutput</strong> components.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data write model</td>
<td>Drop-down list of available data write modes.</td>
</tr>
<tr>
<td>Keys management</td>
<td>Drop-down list of available keys management modes.</td>
</tr>
<tr>
<td>• Keep all</td>
<td>writes all the data received to the cache memory.</td>
</tr>
<tr>
<td>• Keep first</td>
<td>writes only the first record to the cache memory if multiple records received have the same key value.</td>
</tr>
<tr>
<td>Append</td>
<td>Selected by default, this option is designed to append data to the memory in case an iterator exists in the same subjob. If it is unchecked, <strong>tHashOutput</strong> will clear the memory before loading data to it.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>If <strong>Link with a tHashOutput</strong> is selected, this check box will be hidden but is always enabled.</td>
</tr>
</tbody>
</table>

### Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it. For further information about variables, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>

### Usage

| Usage rule | This component writes data to the cache memory and is closely related to **tHashInput**. Together, these twin |
Related scenarios

For related scenarios, see:

- **Scenario 1:** Reading data from the cache memory for high-speed data access on page 1398.
- **Scenario 2:** Clearing the memory before loading data to it in case an iterator exists in the same subjob on page 1402.
tHBaseClose

Closes an HBase connection you have established in your Job.
tHBaseClose closes an active connection to an HBase database.

tHBaseClose Standard properties

These properties are used to configure tHBaseClose running in the Standard Job framework. The Standard tHBaseClose component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

| Component list | Select the tHBaseConnection component in the list if more than one connection are planned for the current Job. |

Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component is to be used along with HBase components, especially with tHBaseConnection. |

| Prerequisites | Before starting, ensure that you have met the Loopback IP prerequisites expected by your database. The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The |
following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the `Job Run VM arguments` area of the `Run/Debug` view in the `[Preferences]` dialog box in the `Window` menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

**Related scenario**

For a scenario in which **tHBaseClose** is used, see Scenario: Exchanging customer data with HBase on page 1424.
tHBaseConnection

Establishes an HBase connection to be reused by other HBase components in your Job.
tHBaseConnection opens a connection to an HBase database.

**tHBaseConnection Standard properties**

These properties are used to configure tHBaseConnection running in the Standard Job framework.
The Standard tHBaseConnection component belongs to the Big Data and the Databases families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Built-in: no property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>- Repository: select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.
In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the Import from existing version option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

Note:

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see Connecting to a custom Hadoop distribution on page 1589.

<table>
<thead>
<tr>
<th>HBase version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop version of the distribution</td>
<td>This list is displayed only when you have selected Custom from the distribution list to connect to a cluster not yet officially supported by the Studio. In this situation, you need to select the Hadoop version of this custom cluster, that is to say, Hadoop 1 or Hadoop 2.</td>
</tr>
<tr>
<td>Zookeeper quorum</td>
<td>Type in the name or the URL of the Zookeeper service you use to coordinate the transaction between your Studio and your database. Note that when you configure the Zookeeper, you might need to explicitly set the zookeeper.znode.parent property to define the path to the root znode that contains all the znodes created and used by your database; then select the Set Zookeeper znode parent check box to define this property.</td>
</tr>
<tr>
<td>Zookeeper client port</td>
<td>Type in the number of the client listening port of the Zookeeper service you are using.</td>
</tr>
</tbody>
</table>
Inspect the classpath for configurations

Select this check box to allow the component to check the configuration files in the directory you have set with the $HADOOP_CONF_DIR variable and directly read parameters from these files in this directory. This feature allows you to easily change the Hadoop configuration for the component to switch between different environments, for example, from a test environment to a production environment.

In this situation, the fields or options used to configure Hadoop connection and/or Kerberos security are hidden.

If you want to use certain parameters such as the Kerberos parameters but these parameters are not included in these Hadoop configuration files, you need to create a file called talend-site.xml and put this file into the same directory defined with $HADOOP_CONF_DIR. This talend-site.xml file should read as follows:

<!-- Put site-specific property overrides in this file. -->
<configuration>
    <property>
        <name>talend.kerberos.authentication</name>
        <value>kinit</value>
        <description>Set the Kerberos authentication method to use. Valid values are: kinit or keytab. </description>
    </property>
    <property>
        <name>talend.kerberos.keytab.principal</name>
        <value>user@BIGDATA.COM</value>
        <description>Set the keytab's principal name. </description>
    </property>
    <property>
        <name>talend.kerberos.keytab.path</name>
        <value>/kdc/user.keytab</value>
        <description>Set the keytab's path. </description>
    </property>
    <property>
        <name>talend.encryption</name>
        <value>none</value>
        <description>Set the encryption method to use. Valid values are: none or ssl. </description>
    </property>
    <property>
        <name>talend.ssl.trustStore.path</name>
        <value>ssl</value>
    </property>
</configuration>
<description> Set SSL trust store path. </description>
</property>

<property>
  <name>talend.s
  sl.trustStore.password</name>
  <value>ssl</value>
  <description> Set SSL trust store password. </description>
</property>

</configuration>

The parameters read from these configuration files override the default ones used by the Studio. When a parameter does not exist in these configuration files, the default one is used.

**Use kerberos authentication**

If the database to be used is running with Kerberos security, select this check box, then, enter the principal names in the displayed fields. You should be able to find the information in the `hbase-site.xml` file of the cluster to be used.

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

If you need to use a Kerberos keytab file to log in, select **Use a keytab to authenticate**. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used.

**Advanced settings**

| Properties                      | If you need to use custom configuration for your HBase, complete this table with the property or properties |
to be customized. Then at runtime, the customized property or properties will override those corresponding ones defined earlier for your HBase.

For example, you need to define the value of the `dfs.replication` property as 1 for the HBase configuration. Then you need to add one row to this table using the plus button and type in the name and the value of this property in this row.

| tStatCatcher Statistics | Select this check box to collect the log data at a component level. |

### Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. |

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | This component is generally used with other HBase components, particularly **tHBaseClose**. |

### Prerequisites

Before starting, ensure that you have met the Loopback IP prerequisites expected by your database.

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`. 

1415
• Set the `-Djava.library.path` argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

**Related scenario**

For a scenario in which `tHBaseConnection` is used, see Scenario: Exchanging customer data with HBase on page 1424.
**tHBaseInput**

Reads data from a given HBase database and extracts columns of selection.

HBase is a distributed, column-oriented database that hosts very large, sparsely populated tables on clusters.

**tHBaseInput** extracts columns corresponding to schema definition. Then it passes these columns to the next component via a Main row link.

**HBase filters**

This table presents the HBase filters available in *Talend Studio* and the parameters required by those filters.

<table>
<thead>
<tr>
<th>Filter type</th>
<th>Filter column</th>
<th>Filter family</th>
<th>Filter operation</th>
<th>Filter value</th>
<th>Filter comparator type</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Column Value Filter</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>It compares the values of a given column against the value defined for the <strong>Filter value</strong> parameter. If the filtering condition is met, all columns of the row will be returned.</td>
</tr>
<tr>
<td>Family filter</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>It returns the columns of the family that meets the filtering condition.</td>
</tr>
<tr>
<td>Qualifier filter</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
<td>It returns the columns whose column qualifiers match the filtering condition.</td>
</tr>
<tr>
<td>Column prefix filter</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>It returns all columns of which the qualifiers have the prefix defined for the Filter column parameter.</td>
</tr>
<tr>
<td>Multiple column prefix filter</td>
<td>Yes (Multiple prefixes are separated by comma, for example, id,id_1,id_2)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>It works the same way as a <strong>Column prefix filter</strong> does but allows specifying multiple prefixes.</td>
</tr>
<tr>
<td>Column range filter</td>
<td>Yes (The ends of a range are separated by comma.)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>It allows intra row scanning and returns all matching columns of a scanned row.</td>
</tr>
<tr>
<td>Row filter</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>It filters on row keys and returns all rows that matches the filtering condition.</td>
</tr>
</tbody>
</table>
The use explained above of the listed HBase filters is subject to revisions made by Apache in its Apache HBase project; therefore, in order to fully understand how to use these HBase filters, we recommend reading Apache's HBase documentation.

**tHBaseInput Standard properties**

These properties are used to configure tHBaseInput running in the Standard Job framework.

The Standard tHBaseInput component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

### Basic settings

| Property type | Either Built-In or Repository.  
|---------------|-------------------------|
|               | Built-In: No property data stored centrally.  
|               | Repository: Select the repository file where the properties are stored.  |

Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).
- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the Import from existing version option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

Note:

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see Connecting to a custom Hadoop distribution on page 1589.

| HBase version | Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using. |
| **Hadoop version of the distribution** | This list is displayed only when you have selected **Custom** from the distribution list to connect to a cluster not yet officially supported by the Studio. In this situation, you need to select the Hadoop version of this custom cluster, that is to say, [Hadoop 1](#) or [Hadoop 2](#). |
| **Zookeeper quorum** | Type in the name or the URL of the Zookeeper service you use to coordinate the transaction between your Studio and your database. Note that when you configure the Zookeeper, you might need to explicitly set the `zookeeper.znode.parent` property to define the path to the root znode that contains all the znodes created and used by your database; then select the [Set Zookeeper znode parent](#) check box to define this property. |
| **Zookeeper client port** | Type in the number of the client listening port of the Zookeeper service you are using. |
| **Use kerberos authentication** | If the database to be used is running with Kerberos security, select this check box, then, enter the principal names in the displayed fields. You should be able to find the information in the `hbase-site.xml` file of the cluster to be used. |
| | • If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652. |
| | Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the [Force MapR ticket authentication](#) check box and the [Use Kerberos authentication](#) check box clear, and then MapR should be able to automatically find that ticket on the fly. |
| | If you need to use a Kerberos keytab file to log in, select **Use a keytab to authenticate**. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver. |
| | Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is **user1** and the principal to be used is **guest**; in this situation, ensure that **user1** has the right to read the keytab file to be used. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields. |
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

<table>
<thead>
<tr>
<th><strong>Built-In:</strong> You create and store the schema locally for this component only. Related topic: see <strong>Talend Studio User Guide.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Repository:</strong> You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <strong>Talend Studio User Guide.</strong></td>
</tr>
</tbody>
</table>

### Set table Namespace mappings

Enter the string to be used to construct the mapping between an Apache HBase table and a MapR table. For the valid syntax you can use, see [http://doc.mapr.com/display/MapR40x/Mapping+Table+Namespace+Between+Apache+HBase+Tables+and+MapR+Tables](http://doc.mapr.com/display/MapR40x/Mapping+Table+Namespace+Between+Apache+HBase+Tables+and+MapR+Tables).

### Table name

Type in the name of the table from which you need to extract columns.

### Define a row selection

Select this check box and then in the **Start row** and the **End row** fields, enter the corresponding row keys to specify the range of the rows you want the current component to extract.

Different from the filters you can set using **Is by filter** requiring the loading of all records before filtering the ones to be used, this feature allows you to directly select only the rows to be used.

### Mapping

Complete this table to map the columns of the table to be used with the schema columns you have defined for the data flow to be processed.

### Die on error

Select the check box to stop the execution of the Job when an error occurs.

Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a **Row > Reject** link.
### Advanced settings

<table>
<thead>
<tr>
<th><strong>tStatCatcher Statistics</strong></th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>
| **Properties**             | If you need to use custom configuration for your database, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override the corresponding ones used by the Studio.  

For example, you need to define the value of the `dfs.replication` property as `1` for the database configuration. Then you need to add one row to this table using the plus button and type in the name and the value of this property in this row.  

**Note:** This table is not available when you are using an existing connection by selecting the **Using an existing connection** check box in the **Basic settings** view. |
| **Is by filter**           | Select this check box to use filters to perform fine-grained data selection from your database, such as selection of keys, or values, based on regular expressions.  

Once selecting it, the **Filter** table that is used to define filtering conditions becomes available.  

This feature leverages filters provided by HBase and subject to constraints explained in Apache HBase documentation. Therefore, advanced knowledge of HBase is required to make full use of these filters. |
| **Logical operation**      | Select the operator you need to use to define the logical relation between filters. This available operators are:  

- **And**: every defined filtering conditions must be satisfied. It represents the relationship `FilterList.Operator.MUST_PASS_ALL`  
- **Or**: at least one of the defined filtering conditions must be satisfied. It represents the relationship `FilterList.Operator.MUST_PASS_ONE` |
| **Filter**                 | Click the button under this table to add as many rows as required, each row representing a filter. The parameters you may need to set for a filter are:  

- **Filter type**: the drop-down list presents pre-existing filter types that are already defined by HBase. Select the type of the filter you need to use.  
- **Filter column**: enter the column qualifier on which you need to apply the active filter. This parameter becomes mandatory depending on the type of the filter and of the comparator you are using. For example, it is not used by the **Row Filter** type but is required by the **Single Column Value Filter** type. |
- **Filter family**: enter the column family on which you need to apply the active filter. This parameter becomes mandatory depending on the type of the filter and of the comparator you are using. For example, it is not used by the Row Filter type but is required by the Single Column Value Filter type.

- **Filter operation**: select from the drop-down list the operation to be used for the active filter.

- **Filter Value**: enter the value on which you want to use the operator selected from the Filter operation drop-down list.

- **Filter comparator type**: select the type of the comparator to be combined with the filter you are using.

Depending on the Filter type you are using, some or each of the parameters become mandatory. For further information, see HBase filters on page 1417

### Retrieve timestamps

Select this check box to load the timestamps of an HBase column into the data flow.

- **Retrieve from an HBase column**: select the HBase column which is tracked for changes in order to retrieve its corresponding timestamps.

- **Write to a schema column**: select the column you have defined in the schema to store the retrieved timestamps.

The type of this column must be **Long**.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

| Usage rule | This component is a start component of a Job and always needs an output link. |
Prerequisites

Before starting, ensure that you have met the Loopback IP prerequisites expected by your database.

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

Scenario: Exchanging customer data with HBase

This scenario applies only to a Talend solution with Big Data.

In this scenario, a six-component Job is used to exchange customer data with a given HBase.
The six components are:

- **tHBaseConnection**: creates a connection to your HBase database.
- **tFixedFlowInput**: creates the data to be written into your HBase. In the real use case, this component could be replaced by the other input components like *tFileInputDelimited*.
- **tHBaseOutput**: writes the data it receives from the preceding component into your HBase.
- **tHBaseInput**: extracts the columns of interest from your HBase.
- **tLogRow**: presents the execution result.
- **tHBaseClose**: closes the transaction.

To replicate this scenario, proceed as the following sections illustrate.

**Note:**
Before starting the replication, your Hbase and Zookeeper service should have been correctly installed and well configured. This scenario explains only how to use Talend solution to make data transaction with a given HBase.

**Dropping and linking the components**

**About this task**
To do this, proceed as follows:

**Procedure**

1. Drop *tHBaseConnection, tFixedFlowInput, tHBaseOutput, tHBaseInput, tLogRow* and **tHBaseClose** from Palette onto the Design workspace.
2. Right-click **tHBaseConnection** to open its contextual menu and select the **Trigger > On Subjob Ok** link from this menu to connect this component to **tFixedFlowInput**.
3. Do the same to create the **OnSubjobOk** link from **tFixedFlowInput** to **tHBaseInput** and then to **tHBaseClose**.

4. Right-click **tFixedFlowInput** and select the **Row > Main** link to connect this component to **tHBaseOutput**.

5. Do the same to create the **Main** link from **tHBaseInput** to **tLogrow**.

### Results

The components to be used in this scenario are all placed and linked. Then you need continue to configure them successively.

### Configuring the connection

#### About this task

To configure the connection to your Zookeeper service and thus to the HBase of interest, proceed as follows:

#### Procedure

1. On the Design workspace of your Studio, double-click the **tHBaseConnection** component to open its **Component** view.

2. Select **Hortonworks Data Platform 1.0** from the **HBase version** list.

3. In the **Zookeeper quorum** field, type in the name or the URL of the Zookeeper service you are using. In this example, the name of the service in use is `hbase`.

4. In the **Zookeeper client port** field, type in the number of client listening port. In this example, it is 2181.

5. If the Zookeeper znode parent location has been defined in the Hadoop cluster you are connecting to, you need to select the **Set zookeeper znode parent** check box and enter the value of this property in the field that is displayed.

### Configuring the process of writing data into the HBase

#### About this task

To do this, proceed as follows:
**Procedure**

1. On the Design workspace, double-click the tFixedFlowInput component to open its Component view.

![tFixedFlowInput component](image)

- **Basic settings**
  - **Schema**: Edit schema
  - **Number of rows**: 1
  - **Mode**
  - Use Single Table
  - Use Inline Table
  - Use Inline Content (delimited file)
  - **Row Separator**: "," 
  - **Field Separator**: ";"
  - **Content**:
    - 1;Albert;23
    - 2;Alexandre;24
    - 3;Alfred-Hubert;22
    - 4;Andre;40
    - 5;Didier;28
    - 6;Anthony;35
    - 7;Artus;32
    - 8;Catherine;34

2. In this view, click the three-dot button next to **Edit schema** to open the schema editor.

![Edit schema](image)

3. Click the plus button three times to add three rows and in the **Column** column, rename the three rows respectively as: id, name and age.

4. In the **Type** column, click each of these rows and from the drop-down list, select the data type of every row. In this scenario, they are **Integer** for id and age, **String** for name.

5. Click **OK** to validate these changes and accept the propagation prompted by the pop-up dialog box.

6. In the **Mode** area, select the **Use Inline Content (delimited file)** to display the fields for editing.

7. In the **Content** field, type in the delimited data to be written into the HBase, separated with the semicolon ";". In this example, they are:

```
1;Albert;23
2;Alexandre;24
3;Alfred-Hubert;22
4;Andre;40
5;Didier;28
6;Anthony;35
7;Artus;32
8;Catherine;34
```
8. Double-click **tHBaseOutput** to open its **Component** view.

**Note:**

If this component does not have the same schema of the preceding component, a warning icon appears. In this case, click the **Sync columns** button to retrieve the schema from the preceding one and once done, the warning icon disappears.

9. Select the **Use an existing connection** check box and then select the connection you have configured earlier. In this example, it is **tHBaseConnection_1**.

10. In the **Table name** field, type in the name of the table to be created in the HBase. In this example, it is **customer**.

11. In the **Action on table** field, select the action of interest from the drop-down list. In this scenario, select **Drop table if exists and create**. This way, if a table named customer exists already in the HBase, it will be disabled and deleted before creating this current table.

12. Click the **Advanced settings** tab to open the corresponding view.
13. In the **Family parameters** table, add two rows by clicking the plus button, rename them as *family1* and *family2* respectively and then leave the other columns empty. These two column families will be created in the HBase using the default family performance options.

   **Note:**
   The **Family parameters** table is available only when the action you have selected in the **Action on table** field is to create a table in HBase. For further information about this **Family parameters** table, see **tHBaseOutput** on page 1432.

14. In the **Families** table of the **Basic settings** view, enter the family names in the **Family name** column, each corresponding to the column this family contains. In this example, the *id* and the *age* columns belong to *family1* and the *name* column to *family2*.

   **Note:**
   These column families should already exist in the HBase to be connected to; if not, you need to define them in the **Family parameters** table of the **Advanced settings** view for creating them at runtime.

### Configuring the process of extracting data from the HBase

**About this task**
To do this, perform the following operations:

**Procedure**
1. Double-click **tHBaseInput** to open its **Component** view.
2. Select the **Use an existing connection** check box and then select the connection you have configured earlier. In this example, it is `tHBaseConnection_1`.

3. Click the three-dot button next to **Edit schema** to open the schema editor.

4. Click the plus button three times to add three rows and rename them as `id`, `name` and `age` respectively in the **Column** column. This means that you extract these three columns from the HBase.

5. Select the types for each of the three columns. In this example, **Integer** for `id` and `age`, **String** for `name`.

6. Click **OK** to validate these changes and accept the propagation prompted by the pop-up dialog box.

7. In the **Table name** field, type in the table from which you extract the columns of interest. In this scenario, the table is `customer`.

8. In the **Mapping** table, the **Column** column has been already filled automatically since the schema was defined, so simply enter the name of every family in the **Column family** column, each corresponding to the column it contains.

9. Double-click `tHBaseClose` to open its **Component** view.
In the **Component List** field, select the connection you need to close. In this example, this connection is **tHBaseConnection_1**.

### Executing the Job

To execute this Job, press **F6**.

Once done, the **Run** view is opened automatically, where you can check the execution result.

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Didier</td>
<td>20</td>
</tr>
<tr>
<td>Emile</td>
<td>32</td>
</tr>
<tr>
<td>Alfred-Hubert</td>
<td>22</td>
</tr>
<tr>
<td>Charles</td>
<td>21</td>
</tr>
<tr>
<td>Albert</td>
<td>23</td>
</tr>
<tr>
<td>Christian</td>
<td>67</td>
</tr>
<tr>
<td>Artus</td>
<td>32</td>
</tr>
<tr>
<td>Christophe</td>
<td>36</td>
</tr>
<tr>
<td>Catherine</td>
<td>34</td>
</tr>
<tr>
<td>Elisabeth</td>
<td>58</td>
</tr>
<tr>
<td>Danielle</td>
<td>54</td>
</tr>
<tr>
<td>Anthony</td>
<td>35</td>
</tr>
<tr>
<td>Alexandre</td>
<td>24</td>
</tr>
<tr>
<td>Andre</td>
<td>40</td>
</tr>
</tbody>
</table>

These columns of interest are extracted and you can process them according to your needs.

Login to your HBase database, you can check the **customer** table this Job has created.
tHBaseOutput

Writes columns of data into a given HBase database.

tHBaseOutput receives data from its preceding component, creates a table in a given HBase database and writes the received data into this HBase table.

**tHBaseOutput Standard properties**

These properties are used to configure tHBaseOutput running in the Standard Job framework.
The Standard tHBaseOutput component belongs to the Big Data and the Databases families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file where the properties are stored.</td>
</tr>
</tbody>
</table>

![Icon](image)

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this **Distribution** drop-down list, the **Microsoft HD Insight** option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select **Amazon EMR**, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).
• The **Custom** option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by **Talend**.

1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend community** have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration list** and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend and its community** provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom Hadoop distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th>HBase version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop version of the distribution</td>
<td>This list is displayed only when you have selected <strong>Custom</strong> from the distribution list to connect to a cluster not yet officially supported by the Studio. In this</td>
</tr>
</tbody>
</table>
### Zookeeper quorum

Type in the name or the URL of the Zookeeper service you use to coordinate the transaction between your Studio and your database. Note that when you configure the Zookeeper, you might need to explicitly set the `zookeeper.znode.parent` property to define the path to the root znode that contains all the znodes created and used by your database; then select the **Set Zookeeper znode parent** check box to define this property.

### Zookeeper client port

Type in the number of the client listening port of the Zookeeper service you are using.

### Use kerberos authentication

If the database to be used is running with Kerberos security, select this check box, then, enter the principal names in the displayed fields. You should be able to find the information in the `hbase-site.xml` file of the cluster to be used.

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in [Connecting to a security-enabled MapR](#) on page 1652.

  Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

If you need to use a Kerberos keytab file to log in, select **Use a keytab to authenticate**. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to *Built-in* for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Built-In
You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](https://help.talend.com).

### Repository
You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](https://help.talend.com).

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### Set table Namespace mappings
Enter the string to be used to construct the mapping between an Apache HBase table and a MapR table.

For the valid syntax you can use, see [http://doc.mapr.com/display/MapR40x/Mapping+Table+Namespace+Between+Apache+HBase+Tables+and+MapR+Tables](http://doc.mapr.com/display/MapR40x/Mapping+Table+Namespace+Between+Apache+HBase+Tables+and+MapR+Tables).

### Table name
Type in the name of the HBase table you need create.

### Action on table
Select the action you need to take for creating an HBase table.

### Custom Row Key
Select this check box to use the customized row keys. Once selected, the corresponding field appears. Then type in the user-defined row key to index the rows of the HBase table being created.

For example, you can type in "France"+Numeric.sequence("s1",1,1) to produce the row key series: France1, France2, France3 and so on.

### Families
Complete this table to map the columns of the table to be used with the schema columns you have defined for the data flow to be processed.

The **Column** column of this table is automatically filled once you have defined the schema; in the **Family name** column, enter the column families you want to create or use to group the columns in the **Column** column. For
further information about a column family, see Apache documentation at Column families.

<table>
<thead>
<tr>
<th>Custom timestamp column</th>
<th>Select a Long column from your schema to provide timestamps for the HBase columns to be created or updated by tHBaseOutput.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die on error</td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>Use batch mode</th>
<th>Select this check box to activate the batch mode for data processing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>Specify the number of records to be processed in each batch.</td>
</tr>
<tr>
<td></td>
<td>This field appears only when the Use batch mode check box is selected.</td>
</tr>
</tbody>
</table>
| Properties     | If you need to use custom configuration for your database, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override the corresponding ones used by the Studio. For example, you need to define the value of the dfs.replication property as 1 for the database configuration. Then you need to add one row to this table using the plus button and type in the name and the value of this property in this row.  

#### Note:
This table is not available when you are using an existing connection by selecting the Using an existing connection check box in the Basic settings view.

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>
| Family parameters       | Type in the names and, when needs be, the custom performance options of the column families to be created. These options are all attributes defined by the HBase data model, so for further explanation about these options, see Apache’s HBase documentation.  

#### Note:
The parameter Compression type allows you to select the format for output data compression.

### Global Variables

| Global Variables | NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. |
**ERROR_MESSAGE:** The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is normally an end component of a Job and always needs an input link.</th>
</tr>
</thead>
</table>

### Prerequisites

Before starting, ensure that you have met the Loopback IP prerequisites expected by your database.

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

  Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the **Preferences** dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the **Data viewer** to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.
Related scenario

For related scenario to the Standard version of **tHBaseOutput**, see **Scenario: Exchanging customer data with HBase** on page 1424.
tHCatalogInput

Reads data from an HCatalog managed Hive database and send data to the component that follows.

The tHCatalogInput component reads data from the specified HCatalog managed database and sends data in the data flow to the console or to a specified local file by connecting this component to a proper component.

**tHCatalogInput Standard properties**

These properties are used to configure tHCatalogInput running in the Standard Job framework.

The Standard tHCatalogInput component belongs to the Big Data family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in:</strong> No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to Built-in for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in:** The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide.

**Repository:** The schema already exists and is stored in the Repository, hence can be reused in various projects and Job designs. Related topic: see Talend Studio User Guide.
Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones require specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**
In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see *Connecting to a custom Hadoop distribution* on page 1589.

<table>
<thead>
<tr>
<th><strong>HCatalog version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Templeton hostname</strong></td>
<td>Fill this field with the URL of Templeton Webservice. <strong>Note:</strong> Templeton is a webservice API for HCatalog. It has been renamed to WebHCat by the Apache community. This service facilitates the access to HCatalog and the related Hadoop elements such as Pig. For further information about Templeton (WebHCat), see <a href="https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat">https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat</a>.</td>
</tr>
<tr>
<td><strong>Templeton port</strong></td>
<td>Fill this field with the port of URL of Templeton Webservice. By default, this value is 50111. <strong>Note:</strong> Templeton is a webservice API for HCatalog. It has been renamed to WebHCat by the Apache community. This service facilitates the access to HCatalog and the related Hadoop elements such as Pig. For further information about Templeton (WebHCat), see <a href="https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat">https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat</a>.</td>
</tr>
</tbody>
</table>
| **Use kerberos authentication** | If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.  
  - If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in *Connecting to a security-enabled MapR* on page 1652.  
  Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the Force MapR ticket authentication check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly. |
This check box is available depending on the Hadoop distribution you are connecting to.

**Use a keytab to authenticate**

Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used.

**Database**

The database in which the HCatalog managed tables are placed. By default, this database is the Hive one named `default`.

**Table**

Fill this field to operate on one or multiple tables in the specified database.

**Partition**

Fill this field to specify one or more partitions for the partition operation on a specified table. When you specify multiple partitions, use commas to separate every two partitions and use double quotation marks to quote the partition string.

If you are reading a non-partitioned table, leave this field empty.

*Note:*

For further information about **Partition**, see [https://cwiki.apache.org/Hive/](https://cwiki.apache.org/Hive/).

**Username**

Fill this field with the username for the Hive database authentication.

**Die on error**

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

**Advanced settings**

| **Row separator** | The separator used to identify the end of a row. |
| **Field separator** | Enter character, string or regular expression to separate fields for the transferred data. |
| **Custom encoding** | Select the encoding from the list or select **Custom** and define it manually. This field is compulsory for database data handling. |
Talend Studio uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on http://hadoop.apache.org/docs and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the hdfs-default.xml file of your distribution, such as http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml.

- Apache also provides a page to list the Hive-related properties: https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties.

### Hadoop properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Talend Studio</td>
<td>Uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.</td>
</tr>
</tbody>
</table>

- Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on http://hadoop.apache.org/docs and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the hdfs-default.xml file of your distribution, such as http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml.

- Apache also provides a page to list the Hive-related properties: https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties.

### Retrieve the HCatalog logs

- Select this check box to retrieve log files generated during HCatalog operations.

### Standard Output Folder

- Fill this field with the path to which log files are stored.

  **Note:**

  This field is enabled only when you selected Retrieve the HCatalog logs check box.

### Error Output Folder

- Fill this field with the path to which error log files are stored.

  **Note:**

  This field is enabled only when you selected Retrieve the HCatalog logs check box.

### tStatCatcher Statistics

- Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is commonly used as the starting component in a Job. HCatalog is built on top of the Hive metastore to provide read and write interface for Pig and MapReduce, so that the latter systems can use the metadata of Hive to easily read and write data in HDFS. For further information, see Apache documentation about HCatalog: <a href="https://cwiki.apache.org/confluence/display/Hive/HCatalog">https://cwiki.apache.org/confluence/display/Hive/HCatalog</a>.</th>
</tr>
</thead>
</table>

Prerequisites

<table>
<thead>
<tr>
<th>Prerequisites</th>
<th>The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR's documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: <a href="http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr">http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr</a>. Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.</td>
<td></td>
</tr>
<tr>
<td>2. Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR. For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.</td>
<td></td>
</tr>
</tbody>
</table>

Limitation

| Limitation | When Use kerberos authentication is selected, the component cannot work with IBM JVM. |
Related scenario

For a related scenario, see *Scenario: HCatalog table management on Hortonworks Data Platform* on page 1459.
tHCatalogLoad

Reads data directly from HDFS and writes this data into an established HCatalog managed table.

tHCatalogLoad Standard properties

These properties are used to configure tHCatalogLoad running in the Standard Job framework.

The Standard tHCatalogLoad component belongs to the Big Data family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

Distribution

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

  1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

  2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.
In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this [Hadoop configuration](#) list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th><strong>HCatalog version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Templeton hostname</strong></td>
<td>Fill this field with the URL of Templeton Webservice. <strong>Note:</strong> Templeton is a webservice API for HCatalog. It has been renamed to WebHCat by the Apache community. This service facilitates the access to HCatalog and the related Hadoop elements such as Pig. For further information about Templeton (WebHCat), see <a href="https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat">https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat</a>.</td>
</tr>
<tr>
<td><strong>Templeton port</strong></td>
<td>Fill this field with the port of URL of Templeton Webservice. By default, this value is 50111. <strong>Note:</strong> Templeton is a webservice API for HCatalog. It has been renamed to WebHCat by the Apache community. This service facilitates the</td>
</tr>
</tbody>
</table>
access to HCatalog and the related Hadoop elements such as Pig. For further information about Templeton (WebHCat), see [https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat](https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat).

**Use kerberos authentication**

If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

This check box is available depending on the Hadoop distribution you are connecting to.

**Use a keytab to authenticate**

Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used.

**Database**

Enter the name of the database you need to write data in. This database must already exist.

**Table**

Enter the name of the table you need to write data in. This table must already exist.

**Partition**

Fill this field to specify one or more partitions for the partition operation on the specified table. When you specify multiple partitions, use commas to separate every two partitions and use double quotation marks to quote the partition string.

If you are reading a non-partitioned table, leave this field empty.
**Note:**
For further information about **Partition**, see https://cwiki.apache.org/Hive/.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>Fill this field with the username for the DB authentication.</td>
</tr>
<tr>
<td>File location</td>
<td>Enter the absolute path pointing to the HDFS location from which data is read.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve the HCatalog logs</td>
<td>Select this check box to retrieve log files generated during HCatalog operations.</td>
</tr>
<tr>
<td>Standard Output Folder</td>
<td>Fill this field with the path to which log files are stored.</td>
</tr>
</tbody>
</table>

**Note:**
This field is enabled only when you selected **Retrieve the HCatalog logs** check box.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Output Folder</td>
<td>Fill this field with the path to which error log files are stored.</td>
</tr>
</tbody>
</table>

**Note:**
This field is enabled only when you selected **Retrieve the HCatalog logs** check box.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Variables</td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>
Usage

Usage rule

This component can be used in a single-component Subjob.

HCatalog is built on top of the Hive metastore to provide read and write interface for Pig and MapReduce, so that the latter systems can use the metadata of Hive to easily read and write data in HDFS.

For further information, see Apache documentation about HCatalog: https://cwiki.apache.org/confluence/display/Hive/HCatalog.

Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

Limitation

When Use kerberos authentication is selected, the component cannot work with IBM JVM.

Related scenario

For a related scenario, see Scenario: HCatalog table management on Hortonworks Data Platform on page 1459.
tHCatalogOperation

Prepares the HCatalog managed database/table/partition to be processed.

tHCatalogOperation manages the data stored in HCatalog managed Hive database/table/partition.

**tHCatalogOperation Standard properties**

These properties are used to configure tHCatalogOperation running in the Standard Job framework. The Standard tHCatalogOperation component belongs to the Big Data family. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally. <strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the **Microsoft HD Insight** option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select **Amazon EMR**, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The **Custom** option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

  1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

  2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.
In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the Import from existing version option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

Note:
In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see Connecting to a custom Hadoop distribution on page 1589.

<table>
<thead>
<tr>
<th>HCatalog version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Templeton hostname</td>
<td>Fill this field with the URL of Templeton Webservice. <strong>Note:</strong> Templeton is a webservice API for HCatalog. It has been renamed to WebHCat by the Apache community. This service facilitates the access to HCatalog and the related Hadoop elements such as Pig. For further information about Templeton (WebHCat), see <a href="https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat">https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat</a>.</td>
</tr>
<tr>
<td>Templeton port</td>
<td>Fill this field with the port of URL of Templeton Webservice. By default, the value for this field is 50111. <strong>Note:</strong> Templeton is a webservice API for HCatalog. It has been renamed to WebHCat by the Apache community. This service facilitates the...</td>
</tr>
</tbody>
</table>
access to HCatalog and the related Hadoop elements such as Pig. For further information about Templeton (WebHCat), see [https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat](https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat).

### Use kerberos authentication

If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in [Connecting to a security-enabled MapR](#) on page 1652.

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

This check box is available depending on the Hadoop distribution you are connecting to.

### Use a keytab to authenticate

Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is *user1* and the principal to be used is *guest*; in this situation, ensure that *user1* has the right to read the keytab file to be used.

### Operation on

Select an object from the list for the DB operation as follows:

- **Database**: The HCatalog managed database in HDFS.
- **Table**: The HCatalog managed table in HDFS.
- **Partition**: The partition specified by the user.

### Operation

Select an action from the list for the DB operation. For further information about the DB operation in HDFS, see [https://cwiki.apache.org/Hive/](https://cwiki.apache.org/Hive/).

### Create the table only if it doesn’t exist already

Select this check box to avoid creating duplicate table when you create a table.
| **Database** | Fill this field with the name of the database in which the HCatalog managed tables are placed. |
| **Table** | Fill this field to operate on one or multiple tables in a database or on a specified HDFS location.  
[i] **Note:** This field is enabled only when you have selected **Table** from the **Operation on** list. For further information about the operation on **Table**, see [https://cwiki.apache.org/Hive/](https://cwiki.apache.org/Hive/). |
| **Partition** | Fill this field to specify one or more partitions for the partition operation on a specified table. When you specify multiple partitions, use comma to separate every two partitions and use double quotation marks to quote the partition string.  
If you are reading a non-partitioned table, leave this field empty.  
[i] **Note:** This field is enabled only when you select **Partition** from the **Operation on** list. For further information about the operation on **Partition**, see [https://cwiki.apache.org/Hive/](https://cwiki.apache.org/Hive/). |
| **Username** | Fill this field with the username for the DB authentication. |
| **Database location** | Fill this field with the location of the database file in HDFS.  
[i] **Note:** This field is enabled only when you select **Database** from the **Operation on** list. |
| **Database description** | The description for the database to be created.  
[i] **Note:** This field is enabled only when you select **Database** from the **Operation on** list. |
| **Create an external table** | Select this field to create an external table in an alternative path defined in the **Set HDFS location** field in the **Advanced settings** view. For further information about creating external table, see [https://cwiki.apache.org/Hive/](https://cwiki.apache.org/Hive/).  
[i] **Note:** This check box is enabled only when you select **Table** from the **Operation on** list and **Create/Drop and create/Drop if exist and create** from the **Operation** list. |
<table>
<thead>
<tr>
<th>Format</th>
<th>Select a file format from the list to specify the format of the external table you want to create:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>TEXTFILE</strong>: Plain text files. <strong>RCFILE</strong>: Record Columnar files. For further information about RCFILE, see <a href="http://hive.apache.org/javadocs/r0.10.0/api/org/apache/hadoop/hive/ql/io/RCFile.html">http://hive.apache.org/javadocs/r0.10.0/api/org/apache/hadoop/hive/ql/io/RCFile.html</a>.</td>
</tr>
<tr>
<td>Note:</td>
<td><strong>RCFILE</strong> is only available starting with Hive 0.6.0. This list is enabled only when you select <strong>Table</strong> from the <strong>Operation on</strong> list and <strong>Create/Drop and create/Drop if exist and create</strong> from the <strong>Operation</strong> list.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: This check box is enabled only when you select <strong>Table</strong> from the <strong>Operation on</strong> list and <strong>Create/Drop and create/Drop if exist and create</strong> from the <strong>Operation</strong> list. You must follow the rules of using partition schema in HCatalog managed tables. For more information about the rules in using partition schema, see <a href="https://cwiki.apache.org/confluence/display/Hive/HCatalog">https://cwiki.apache.org/confluence/display/Hive/HCatalog</a>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Set partitions</th>
<th>Select this check box to set the partition schema by clicking the <strong>Edit schema</strong> to the right of <strong>Set partitions</strong> check box. The partition schema is either built-in or remote in the Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>This check box is enabled only when you select <strong>Table</strong> from the <strong>Operation on</strong> list and <strong>Create/Drop and create/Drop if exist and create</strong> from the <strong>Operation</strong> list. You must follow the rules of using partition schema in HCatalog managed tables. For more information about the rules in using partition schema, see <a href="https://cwiki.apache.org/confluence/display/Hive/HCatalog">https://cwiki.apache.org/confluence/display/Hive/HCatalog</a>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Set the user group to use</th>
<th>Select this check box to specify the user group.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>This check box is enabled only when you select <strong>Drop/Drop if exist/Drop and create/Drop if exist and create</strong> from the <strong>Operation</strong> list. By default, the value for this field is <strong>root</strong>. For more information about the user group in the server, contact your system administrator.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Select a clause when you drop a database.</th>
</tr>
</thead>
</table>
| Note:  | This list is enabled only when you select **Database** from the **Operation on** list and **Drop/Drop if exist/Drop and create/Drop if exist and create** from the **Operation** list. For
more information about **Drop** operation on database, see [https://cwiki.apache.org/Hive/](https://cwiki.apache.org/Hive/).

| **Set the permissions to use** | Select this check box to specify the permissions needed by the operation you select from the **Operation** list.  

**Note:** 
This check box is enabled only when you select **Drop/Drop if exist/Drop and create/Drop if exist and create** from the **Operation** list. By default, the value for this field is `rwxrwx-r-x`. For more information on user permissions, contact your system administrator. |
|---|---|
| **Set File location** | Enter the directory in which partitioned data is stored.  

**Note:**  
This check box is enabled only when you select **Partition** from the **Operation on** list and **Create/Drop and create/Drop if exist and create** from the **Operation** list. For further information about storing partitioned data in HDFS, see [https://cwiki.apache.org/Hive/](https://cwiki.apache.org/Hive/). |
| **Die on error** | This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows. |

**Advanced settings**

| **Comment** | Fill this field with the comment for the table you want to create.  

**Note:**  
This field is enabled only when you select **Table** from the **Operation on** list and **Create/Drop and create/Drop if exist and create** from the **Operation** list in the **Basic settings** view. |
| **Set HDFS location** | Select this check box to specify an HDFS location to which the table you want to create is saved. Deselect it to save the table you want to create in the warehouse directory defined in the key `hive.metastore.warehouse.dir` in Hive configuration file `hive-site.xml`.  

**Note:**  
This check box is enabled only when you select **Table** from the **Operation on** list and **Create/Drop and create/Drop if exist and create** from the **Operation** list in the **Basic settings** view. For further information about saving data in HDFS, see [https://cwiki.apache.org/Hive/](https://cwiki.apache.org/Hive/). |
| **Set row format(terminated by)** | Select this check box to use and define the row formats when you want to create a table: |
**Field**: Select this check box to use Field as the row format. The default value for this field is "\u0001". You can also specify a customized char in this field.

**Collection Item**: Select this check box to use Collection Item as the row format. The default value for this field is "\u0002". You can also specify a customized char in this field.

**Map Key**: Select this check box to use Map Key as the row format. The default value for this field is "\u0003". You can also specify a customized char in this field.

**Line**: Select this check box to use Line as the row format. The default value for this field is '\n'. You can also specify a customized char in this field.

**Properties**

Click [+] to add one or more lines to define table properties. The table properties allow you to tag the table definition with your own metadata key/value pairs. Make sure that values in both Key row and Value row must be quoted in double quotation marks.

**Retrieve the HCatalog logs**

Select this check box to retrieve log files generated during HCatalog operations.

**Standard Output Folder**

Browse to, or enter the directory where the log files are stored.

**Error Output Folder**

Browse to, or enter the directory where the error log files are stored.
tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Global Variables

Global Variables

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

Usage

Usage rule

This component is commonly used in a single-component subjob.

HCatalog is built on top of the Hive metastore to provide read and write interface for Pig and MapReduce, so that the latter systems can use the metadata of Hive to easily read and write data in HDFS.

For further information, see Apache documentation about HCatalog: [https://cwiki.apache.org/confluence/display/Hive/HCatalog](https://cwiki.apache.org/confluence/display/Hive/HCatalog).

Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with [Talend Studio](#). The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR's documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

  Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the
**Run/Debug** view in the [Preferences] dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the **Data viewer** to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

| Limitation | When **Use kerberos authentication** is selected, the component cannot work with IBM JVM. |

### Scenario: HCatalog table management on Hortonworks Data Platform

This scenario applies only to a Talend solution with Big Data.

This scenario describes a six-component Job that includes the common operations for the HCatalog table management on Hortonworks Data Platform. Sub-sections in this scenario covers DB operations including:

- Creating a table to the database in HDFS;
- Writing data to the HCatalog managed table;
- Writing data to the partitioned table using **tHCatalogLoad**;
- Reading data from the HCatalog managed table;
- Outputting the data read from the table in HDFS.

**Note:**

Knowledge of Hive Data Definition Language and HCatalog Data Definition Language is required. For further information about Hive Data Definition Language, see [https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL). For further information about HCatalog Data Definition Language, see [https://cwiki.apache.org/confluence/display/HCATALOG/Design+Document+-+Java+APIs+for+HCatalog+DDL+Commands](https://cwiki.apache.org/confluence/display/HCATALOG/Design+Document+-+Java+APIs+for+HCatalog+DDL+Commands).

### Setting up the Job

**Procedure**

1. Drop the following components from the **Palette** to the design workspace: **tHCatalogOperation**, **tHCatalogLoad**, **tHCatalogInput**, **tHCatalogOutput**, **tFixedFlowInput**, and **tFileOutputDelimited**.
2. Right-click **tHCatalogOperation** to connect it to **tFixedFlowInput** component using a **Trigger>OnSubjobOk** connection.

3. Right-click **tFixedFlowInput** to connect it to **tHCatalogOutput** using a **Row > Main** connection.

4. Right-click **tFixedFlowInput** to connect it to **tHCatalogLoad** using a **Trigger > OnSubjobOk** connection.

5. Right-click **tHCatalogLoad** to connect it to the **tHCatalogInput** component using a **Trigger > OnSubjobOk** connection.

6. Right-click **tHCatalogInput** to connect it to **tFileOutputDelimited** using a **Row > Main** connection.

**Creating a table in HDFS**

**Procedure**

1. Double-click **tHCatalogOperation** to open its **Basic settings** view.
2. Click **Edit schema** to define the schema for the table to be created.

3. Click [+] to add at least one column to the schema and click **OK** when you finish setting the schema. In this scenario, the columns added to the schema are: `name`, `country` and `age`.

4. Fill the **Templeton hostname** field with URL of the Templeton webservice you are using. In this scenario, fill this field with "192.168.0.131".

5. Fill the **Templeton port** field with the port for **Templeton hostname**. By default, the value for this field is "50111".

6. Select **Table** from the **Operation on** list and **Drop if exist and create** from the **Operation** list to create a table in HDFS.

7. Fill the **Database** field with an existing database name in HDFS. In this scenario, the database name is "talend".
8. Fill the **Table** field with the name of the table to be created. In this scenario, the table name is "Customer".

9. Fill the **Username** field with the username for the DB authentication.

10. Select the **Set the user group to use** check box to specify the user group. The default user group is "root", you need to specify the value for this field according to real practice.

11. Select the **Set the permissions to use** check box to specify the user permission. The default value for this field is "rwxrwxr-x".

12. Select the **Set partitions** check box to enable the partition schema.

13. Click the **Edit schema** button next to the **Set partitions** check box to define the partition schema.

14. Click [+] to add one column to the schema and click **OK** when you finish setting the schema. In this scenario, the column added to the partition schema is: *match_age*.

### Writing data to the existing table

**Procedure**

1. Double-click **tFixedFlowInput** to open its **Basic settings** view.

   ![tFixedFlowInput_1](image)

   - **Basic settings**
     - **Schema**: Built-In
     - **Advanced settings**:
     - **Dynamic settings**:
     - **View**
     - **Documentation**
     - **Validation Rules**
   - **Number of rows**: 8
   - **Mode**:
     - Use Single Table
     - Use Inline Table
   - **Inline Table**
     - | age | country | name |
       |-----|---------|------|
       | "27" | "Tunisia" | "Abu" |
       | "31" | "Senegal" | "Arelia" |
       | "26" | "Morocco" | "Karim" |
       | "33" | "Senegal" | "Dragba" |
       | "27" | "Japan" | "ito" |
       | "26" | "Algeria" | "Zed" |
       | "35" | "Japan" | "Hidetoshi" |
       | "28" | "Lebanon" | "Nasri" |

2. Click **Edit schema** to define a same schema as the one you defined in **tHCatalogOperation**.

3. Fill the **Number of rows** field with integer 8.

4. Select **Use Inline Table** in the **Mode** area.

5. Click [+] to add new lines in the inline table.

6. Double-click **tHCatalogOutput** to open its **Basic settings** view.
7. Click **Sync columns** to retrieve the schema defined in the preceding component.

8. Fill the **NameNode URI** field with the URI to the NameNode. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS.

9. Fill the **File name** field with the HDFS location of the file you write data to. In this scenario, the file location is "/user/hdp/Customer/Customer.csv".

10. Select **Overwrite** from the **Action** list.

11. Fill the **Templeton hostname** field with URL of the Templeton webservice you are using. In this scenario, fill this field with "192.168.0.131".

12. Fill the **Templeton port** field with the port for **Templeton hostname**. By default, the value for this field is "50111".

13. Fill the **Database** field, the **Table** field, the **Username** field with the same value you specified in **tHCatalogOperation**.

14. Fill the **Partition** field with "match_age=27".

15. Fill the **File location** field with the HDFS location to which the table will be saved. In this example, use "hdfs://192.168.0.131:8020/user/hdp/Customer".

---

**Writing data to the partitioned table using tHCatalogLoad**

**Procedure**

1. Double-click **tHCatalogLoad** to open its **Basic settings** view.
2. Fill the **Partition** field with "match_age=26".
3. Do the rest of the settings in the same way as configuring **tHCatalogOperation**.

**Reading data from the table in HDFS**

**Procedure**

1. Double-click **tHCatalogInput** to open its **Basic settings** view.

2. Click **Edit schema** to define the schema of the table to be read from the database.
3. Click [+] to add at least one column to the schema. In this scenario, the columns added to the schema are age and name.

4. Fill the Partition field with "match_age=26".

5. Do the rest of the settings in the same way as configuring tHCatalogOperation.

**Outputting the data read from the table in HDFS to the console**

**Procedure**

1. Double-click tLogRow to open its Basic settings view.

2. Click Sync columns to retrieve the schema defined in the preceding component.

3. Select Table from the Mode area.

**Job execution**

Press CTRL+S to save your Job and F6 to execute it.
The data of the restricted table read from the HDFS is displayed onto the console.

Type in `http://talend-hdp:50075/browseDirectory.jsp?dir=/user/hdp/Customer&namenodeInfoPort=50070` to the address bar of your browser to view the table you created:
Contents of directory /user/hdp/Customer

Goto: /user/hdp/Customer go

Go to parent directory

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Size</th>
<th>Replication</th>
<th>Block Size</th>
<th>Modification Time</th>
<th>Permission</th>
<th>Owner</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer.csv</td>
<td>file</td>
<td>1.04 KB</td>
<td>3</td>
<td>64 MB</td>
<td>2012-04-18 16:16</td>
<td>rw-r--r--</td>
<td>hdp</td>
<td>hdfs</td>
</tr>
</tbody>
</table>

Go back to DFS home

Local logs

Log directory

This is Apache Hadoop release 1.0.2

Click the Customer.csv link to view the content of the table you created.
## tHCatalogOutput

Receives data from its incoming flow and writes this data into an HCatalog managed table.

### tHCatalogOutput Standard properties

These properties are used to configure tHCatalogOutput running in the Standard Job framework.

The Standard tHCatalogOutput component belongs to the Big Data family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

#### Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

#### Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.
You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

### Distribution

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select *Import from existing version* to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select *Import from zip* to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the *Import from existing version* option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient
Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see Connecting to a custom Hadoop distribution on page 1589.

<table>
<thead>
<tr>
<th>HCatalog version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
</table>
| Use kerberos authentication | If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.  
  - If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.  
  
  Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the Force MapR ticket authentication check box and the Use Kerberos authentication check box clear, and then MapR should be able to automatically find that ticket on the fly.  
  
  This check box is available depending on the Hadoop distribution you are connecting to. |
| Use a keytab to authenticate | Select the Use a keytab to authenticate check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the Principal field and the access path to the keytab file itself in the Keytab field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.  
  
  Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is user1 and the principal to be used is guest; in this situation, ensure that user1 has the right to read the keytab file to be used. |
| NameNode URI | Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that |
you have chosen a machine called masternode as the NameNode, then the location is hdfs://masternode:portnumber. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS.

<table>
<thead>
<tr>
<th><strong>File name</strong></th>
<th>Browse to, or enter the location of the file which you write data to. This file is created automatically if it does not exist.</th>
</tr>
</thead>
</table>
| **Action**   | Select a DB operation in HDFS:  
  **Create**: Creates a file with data using the file name defined in the **File Name** field.  
  **Overwrite**: Overwrites the data in the file specified in the **File Name** field.  
  **Append**: Inserts the data into the file specified in the **File Name** field. The specified file is created automatically if it does not exist. |
| **Templeton hostname** | Fill this field with the URL of Templeton Webservice.  
  **Note:**  
  Templeton is a webservice API for HCatalog. It has been renamed to WebHCat by the Apache community. This service facilitates the access to HCatalog and the related Hadoop elements such as Pig. For further information about Templeton (WebHCat), see https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat. |
| **Templeton port** | Fill this field with the port of URL of Templeton Webservice. By default, this value is 50111.  
  **Note:**  
  Templeton is a webservice API for HCatalog. It has been renamed to WebHCat by the Apache community. This service facilitates the access to HCatalog and the related Hadoop elements such as Pig. For further information about Templeton (WebHCat), see https://cwiki.apache.org/confluence/display/Hive/WebHCat+UsingWebHCat. |
| **Database** | Fill this field to specify an existing database in HDFS. |
| **Table** | Fill this field to specify an existing table in HDFS. |
| **Partition** | Fill this field to specify one or more partitions for the partition operation on the specified table. When you specify multiple partitions, use commas to separate every two partitions and use double quotation marks to quote the partition string. |
If you are reading a non-partitioned table, leave this field empty.

**Note:**
For further information about **Partition**, see https://cwiki.apache.org/Hive/.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Username</strong></td>
<td>Fill this field with the username for the DB authentication.</td>
</tr>
<tr>
<td><strong>File location</strong></td>
<td>Fill this field with the path to which source data file is stored.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row separator</strong></td>
<td>The separator used to identify the end of a row.</td>
</tr>
<tr>
<td><strong>Field separator</strong></td>
<td>Enter character, string or regular expression to separate fields for the transferred data.</td>
</tr>
<tr>
<td><strong>Custom encoding</strong></td>
<td>Select the encoding from the list or select Custom and define it manually. This field is compulsory for database data handling.</td>
</tr>
<tr>
<td><strong>Hadoop properties</strong></td>
<td><em>Talend Studio</em> uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.</td>
</tr>
<tr>
<td></td>
<td>• Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.</td>
</tr>
<tr>
<td></td>
<td>For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on <a href="http://hadoop.apache.org/docs">http://hadoop.apache.org/docs</a> and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:</td>
</tr>
<tr>
<td></td>
<td>• Typically, the HDFS-related properties can be found in the <em>hdfs-default.xml</em> file of your distribution, such as <a href="http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml">http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml</a>.</td>
</tr>
<tr>
<td></td>
<td>• Apache also provides a page to list the Hive-related properties: <a href="https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties">https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties</a>.</td>
</tr>
</tbody>
</table>
**Retrieve the HCatalog logs**
Select this check box to retrieve log files generated during HCatalog operations.

**Standard Output Folder**
Browse to, or enter the directory where the log files are stored.

*Note:*
This field is enabled only when you selected **Retrieve the HCatalog logs** check box.

**Error Output Folder**
Browse to, or enter the directory where the error log files are stored.

*Note:*
This field is enabled only when you selected **Retrieve the HCatalog logs** check box.

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

---

### Global Variables

**ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

---

### Usage

**Usage rule**
This component is commonly used together with an input component.

HCatalog is built on top of the Hive metastore to provide read and write interface for Pig and MapReduce, so that the latter systems can use the metadata of Hive to easily read and write data in HDFS.

For further information, see Apache documentation about HCatalog: [https://cwiki.apache.org/confluence/display/Hive/HCatalog](https://cwiki.apache.org/confluence/display/Hive/HCatalog).

**Prerequisites**
The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.
• Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

• Set the `-Djava.library.path` argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

---

**Related scenario**

For a related scenario, see [Scenario: HCatalog table management on Hortonworks Data Platform](#) on page 1459.
tHDFSCompare

Compared two files in HDFS and based on the read-only schema, generates a row flow that presents the comparison information.

tHDFSCompare helps to control the quality of the data processed.

**tHDFSCompare Standard properties**

These properties are used to configure tHDFSCompare running in the Standard Job framework.

The Standard tHDFSCompare component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong> No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong> Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the HDFS connection component from which you want to reuse the connection details already defined.

Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.
1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th><strong>Hadoop version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
</table>

| **Use kerberos authentication** | If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.  
  • If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by |
following the explanation in Connecting to a security-enabled MapR on page 1652.

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

This check box is available depending on the Hadoop distribution you are connecting to.

| Use a keytab to authenticate | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is **user1** and the principal to be used is **guest**; in this situation, ensure that **user1** has the right to read the keytab file to be used. |
| NameNode URI | Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called **masternode** as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a **tLibraryLoad** in the Job to load the library required by the secured WebHDFS. |
| User name | The **User name** field is available when you are not using Kerberos to authenticate. In the **User name** field, enter the login user name for your distribution. If you leave it empty, the user name of the machine hosting the Studio will be used. |
| Group | Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using. |
| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields.

The schema of this component is read-only. You can click **Edit schema** to view the schema. |
<table>
<thead>
<tr>
<th><strong>Comparison mode</strong></th>
<th>Select the mode to be applied on the comparison.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File to compare</strong></td>
<td>Browse, or enter the path to the file in HDFS you need to check for quality control.</td>
</tr>
<tr>
<td><strong>Reference file</strong></td>
<td>Browse, or enter the path to the file in HDFS the comparison is based on.</td>
</tr>
<tr>
<td><strong>If differences detected, display</strong> and <strong>If no differences detected, display</strong></td>
<td>Type in a message to be displayed in the Run console based on the result of the comparison.</td>
</tr>
<tr>
<td><strong>Print to console</strong></td>
<td>Select this check box to display the message in the Run console.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Encoding</strong></th>
<th>Select the encoding from the list or select Custom and define it manually. This field is compulsory for database data handling.</th>
</tr>
</thead>
</table>
| **Hadoop properties** | Talend Studio uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.  
  - Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.  
  For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on http://hadoop.apache.org/docs and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:  
  - Typically, the HDFS-related properties can be found in the hdfs-default.xml file of your distribution, such as http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml.  
  - Apache also provides a page to list the Hive-related properties: https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties. |
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |
### Global Variables

**DIFFERENCE**: the result of the comparison. This is a Flow variable and it returns a boolean.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

**Usage rule**

**tHDFSCompare** can be standalone component or send the information it generates to its following component.

**Dynamic settings**

Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of *Talend Studio*.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\...
hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

| Limitation | JRE 1.6+ is required. |

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tHDFSConnection

Connects to a given HDFS so that the other Hadoop components can reuse the connection it creates to communicate with this HDFS.

tHDFSConnection provides connection to the Hadoop distributed file system (HDFS) of interest at runtime.

**tHDFSConnection Standard properties**

These properties are used to configure tHDFSConnection running in the Standard Job framework.

The Standard tHDFSConnection component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip
The file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th>Hadoop version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspect the classpath for configurations</td>
<td>Select this check box to allow the component to check the configuration files in the directory you have set with the <code>$HADOOP_CONF_DIR</code> variable and directly read parameters from these files in this directory. This feature allows you to easily change the Hadoop configuration for the component to switch between different environments, for example, from a test environment to a production environment. In this situation, the fields or options used to configure Hadoop connection and/or Kerberos security are hidden. If you want to use certain parameters such as the Kerberos parameters but these parameters are not included in these Hadoop configuration files, you need to create a file called <code>talend-site.xml</code> and put this file into</td>
</tr>
</tbody>
</table>
the same directory defined with $HADOOP_CONF_DIR.
This talend-site.xml file should read as follows:

```xml
<!-- Put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>talend.kerberos.authentication</name>
    <value>kinit</value>
    <description>Set the Kerberos authentication method to use. Valid values are: kinit or keytab.</description>
  </property>
  <property>
    <name>talend.kerberos.keytab.principal</name>
    <value>user@BIGDATA.COM</value>
    <description>Set the keytab's principal name.</description>
  </property>
  <property>
    <name>talend.kerberos.keytab.path</name>
    <value>/kdc/user.keytab</value>
    <description>Set the keytab's path.</description>
  </property>
  <property>
    <name>talend.encryption</name>
    <value>none</value>
    <description>Set the encryption method to use. Valid values are: none or ssl.</description>
  </property>
  <property>
    <name>talend.ssl.trustStore.path</name>
    <value>ssl</value>
    <description>Set SSL trust store path.</description>
  </property>
  <property>
    <name>talend.ssl.trustStore.password</name>
    <value>ssl</value>
    <description>Set SSL trust store password.</description>
  </property>
</configuration>
```

The parameters read from these configuration files override the default ones used by the Studio. When a
<table>
<thead>
<tr>
<th><strong>Parameter</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Use kerberos authentication** | If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.  
  - If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in [Connecting to a security-enabled MapR](#) on page 1652.  
  
  Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.  
  
  This check box is available depending on the Hadoop distribution you are connecting to. |
| **Use a keytab to authenticate** | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.  
  
  Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is *user1* and the principal to be used is *guest*; in this situation, ensure that *user1* has the right to read the keytab file to be used. |
| **NameNode URI** | Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called *masternode* as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS. |
| **User name** | User authentication name of HDFS. |
| **Group** | Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using. |
### Hadoop properties

*Talend Studio* uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).
- Apache also provides a page to list the Hive-related properties: [https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties).

### Use datanode hostname

Select the **Use datanode hostname** check box to allow the Job to access datanodes via their hostnames. This actually sets the `dfs.client.use.datanode.hostname` property to `true`.

### Setup HDFS encryption configurations

If the HDFS transparent encryption has been enabled in your cluster, select the **Setup HDFS encryption configurations** check box and in the **HDFS encryption key provider** field that is displayed, enter the location of the KMS proxy.

For further information about the HDFS transparent encryption and its KMS proxy, see Transparent Encryption in HDFS.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

### Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is generally used with other Hadoop components.</th>
</tr>
</thead>
</table>

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the *Job Run VM arguments* area of the *Run/Debug* view in the [Preferences] dialog box in the *Window* menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the *Data viewer* to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

### Limitations

JRE 1.6+ is required.

### Related scenarios

No scenario is available for the Standard version of this component yet.
**tHDFSCopy**

copies a source file or folder into a target directory in HDFS and removes this source if required.

**tHDFSCopy Standard properties**

These properties are used to configure tHDFSCopy running in the Standard Job framework.

The Standard tHDFSCopy component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the HDFS connection component from which you want to reuse the connection details already defined.

Note that when a Job contains the parent Job and the child Job, **Component List** presents only the connection components in the same Job level.

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this **Distribution** drop-down list, the **Microsoft HD Insight** option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select **Amazon EMR**, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The **Custom** option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

  1. Select **Import from existing version** to import an officially supported distribution as base and then add
other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th>Hadoop version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use kerberos authentication</td>
<td>If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.</td>
</tr>
<tr>
<td></td>
<td>• If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in <strong>Connecting to a security-enabled MapR</strong> on page 1652.</td>
</tr>
</tbody>
</table>
Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

This check box is available depending on the Hadoop distribution you are connecting to.

| **Use a keytab to authenticate** | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is *user1* and the principal to be used is *guest*; in this situation, ensure that *user1* has the right to read the keytab file to be used. |
| **NameNode URI** | Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called *masternode* as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS. |
| **User name** | The **User name** field is available when you are not using Kerberos to authenticate. In the **User name** field, enter the login user name for your distribution. If you leave it empty, the user name of the machine hosting the Studio will be used. |
| **Group** | Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using. |
| **Source file or directory** | Browse to, or enter the path pointing to the data to be used in the file system. |
| **Target location** | Browse to, or enter the directory in HDFS to which you need to copy the data. |
| **Rename** | To rename the file or folder copied to the target location, select this check box to display the **New name** field, then, enter the new name. |
### Copy merge
Select this check box to merge the part files generated at the end of a MapReduce computation. Once selecting it, you need to enter the name of the final merged file in the Merge name field.

### Remove source
Select this check box to remove the source file or folder once this source is copied to the target location.

### Override target file (This option does not override the directory)
Select this check box to override the file already existing in the target location. This option does not override the folder.

## Advanced settings

### Hadoop properties
*Talend Studio* uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache's Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).
- Apache also provides a page to list the Hive-related properties: [https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties).

### tStatCatcher Statistics
Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

## Global Variables

### Global Variables
**DESTINATION_FILEPATH**: the destination file path. This is an After variable and it returns a string.

**SOURCE_FILEPATH**: the source file path. This is an After variable and it returns a string.
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

---

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>thHDFSCopy is a standalone component.</th>
</tr>
</thead>
</table>

#### Dynamic settings

Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of *Talend Studio*.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

---

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/](http://www.mapr.com/)
Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the **[Preferences]** dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the **Data viewer** to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

| Limitation       | JRE 1.6+ is required. |

**Related scenario**

Related topic, see **Procedure** on page 985

Related topic, see **Scenario: Iterating on a HDFS directory** on page 1533
tHDFSDelete

Deletes a file located on a given Hadoop distributed file system (HDFS).

**tHDFSDelete Standard properties**

These properties are used to configure tHDFSDelete running in the Standard Job framework.

The Standard tHDFSDelete component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

### Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the HDFS connection component from which you want to reuse the connection details already defined.

Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the **Microsoft HD Insight** option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select **Amazon EMR**, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The **Custom** option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select **Import from existing version** to import an officially supported distribution as base and then add...
other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th><strong>Hadoop version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use kerberos authentication</strong></td>
<td>If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.</td>
</tr>
</tbody>
</table>

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in **Connecting to a security-enabled MapR** on page 1652.
Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

This check box is available depending on the Hadoop distribution you are connecting to.

### Use a keytab to authenticate

Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used.

### NameNode URI

Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called `masternode` as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS.

### User name

User authentication name of HDFS.

### Group

Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using.

### File or Directory Path

Browse to, or enter the path to the file or folder to be deleted on HDFS.

### Advanced settings

#### Hadoop properties

If you need to use custom configuration for the Hadoop of interest, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those corresponding ones defined earlier for the same Hadoop.
For further information about the properties required by Hadoop, see the Hadoop documentation.

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

Global Variables

| DELETE_PATH: the path to the deleted file or folder. This is an After variable and it returns a string. |
| CURRENT_STATUS: the execution result of the component. This is an After variable and it returns a string. |
| ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

| Usage rule | This component is used to compose a single-component Job or Subjob. |

Dynamic settings

| Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on Dynamic settings and context variables, see Talend Studio User Guide.
**Prerequisites**

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

  Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

  For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

**Limitations**

JRE 1.6+ is required.

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tHDFSEexist

Checks whether a file exists in a specific directory in HDFS.

**tHDFSEexist Standard properties**

These properties are used to configure tHDFSEexist running in the Standard Job framework.

The Standard tHDFSEexist component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add
other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

   In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

   Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

   ![Note:](image)

   In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

   For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th>Hadoop version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
</table>
| Use kerberos authentication | If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.  
- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in **Connecting to a security-enabled MapR** on page 1652. |
Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

This check box is available depending on the Hadoop distribution you are connecting to.

**Use a keytab to authenticate**

Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used.

**NameNode URI**

Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called `masternode` as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS.

**User name**

The **User name** field is available when you are not using Kerberos to authenticate. In the **User name** field, enter the login user name for your distribution. If you leave it empty, the user name of the machine hosting the Studio will be used.

**Group**

Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using.

**HDFS directory**

Browse to, or enter the path pointing to the data to be used in the file system.

**File name or relative path**

Enter the name of the file you want to check whether this file exists. Or if needs be, browse to the file or enter the path to the file, relative to the directory you entered in **HDFS directory**.
## Advanced settings

| Hadoop properties | Talend Studio uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.  

- Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.  

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:  

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).  

- Apache also provides a page to list the Hive-related properties: [https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties). |
| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

## Global Variables

| Global Variables | EXISTs: the result of whether a specified file exists. This is a Flow variable and it returns a boolean.  

FILENAME: the name of the file processed. This is an After variable and it returns a string.  

ERROR MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  

For further information about variables, see Talend Studio User Guide. |

| Global Variables | EXISTs: the result of whether a specified file exists. This is a Flow variable and it returns a boolean.  

FILENAME: the name of the file processed. This is an After variable and it returns a string.  

ERROR MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  

For further information about variables, see Talend Studio User Guide. |
### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tHDFSExist is a standalone component.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamic settings</strong></td>
<td>Click the [+ ] button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of <strong>Talend Studio</strong>. The <strong>Dynamic settings</strong> table is available only when the <strong>Use an existing connection</strong> check box is selected in the <strong>Basic settings</strong> view. Once a dynamic parameter is defined, the <strong>Component List</strong> box in the <strong>Basic settings</strong> view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on <strong>Dynamic settings</strong> and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prerequisites</th>
<th>The Hadoop distribution must be properly installed, so as to guarantee the interaction with <strong>Talend Studio</strong>. The following list presents MapR related information for example.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: <a href="http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr">http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr</a>. Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.</td>
</tr>
<tr>
<td></td>
<td>• Set the -Djava.library.path argument, for example, in the <strong>Job Run VM arguments</strong> area of the <strong>Run/Debug</strong> view in the [Preferences] dialog box in the <strong>Window</strong> menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the <strong>Data viewer</strong> to view locally in the Studio the data stored in MapR.</td>
</tr>
</tbody>
</table>
Scenario: Checking the existence of a file in HDFS

This scenario applies only to a Talend solution with Big Data.

In this scenario, the two-component Job checks whether a specific file exists in HDFS and returns a message to indicate the result of the verification.

In the real-world practice, you can take further action to process the file checked according to the verification result, using the other HDFS components provided with the Studio.

Launch the Hadoop distribution in which you want to check the existence of a particular file. Then, proceed as follows:

Linking the components

Procedure

1. In the Integration perspective of Talend Studio, create an empty Job, named hdfsexist_file for example, from the Job Designs node in the Repository tree view.
   For further information about how to create a Job, see the Talend Studio User Guide.
2. Drop tHDFSExist and tMsgBox onto the workspace.
3. Connect them using the Trigger > Run if link.

Configuring the connection to HDFS

Procedure

1. Double-click tHDFSExist to open its Component view.
2. In the **Version** area, select the Hadoop distribution you are connecting to and its version.

3. In the **Connection** area, enter the values of the parameters required to connect to the HDFS.
   In the real-world practice, you may use **tHDFSConnection** to create a connection and reuse it from the current component. For further information, see **tHDFSConnection** on page 1482.

4. In the **HDFS Directory** field, browse to, or enter the path to the folder where the file to be checked is. In this example, browse to `/user/ychen/data/hdfs/out/dest`.

5. In the **File name or relative path** field, enter the name of the file you want to check the existence. For example, **output.csv**.

**Defining the message to be returned**

**Procedure**

1. Double-click **tMsgBox** to open its **Component** view.

2. In the **Title** field, enter the title to be used for the pop-up message box to be created.

3. In the **Buttons** list, select **OK**. This defines the button to be displayed on the message box.

4. In the **Icon** list, select **Icon Information**.

5. In the **Message** field, enter the message you want to displayed once the file checking is done. In this example, enter **"This file does not exist!"**.
Defining the condition

Procedure

1. Click the If link to open the Basic settings view, where you are able to define the condition for checking the existence of this file.

2. In the Condition box, press Ctrl+Space to access the variable list and select the global variable EXISTS. Type an exclamation mark before the variable to negate the meaning of the variable.

Executing the Job

Procedure

Press F6 to execute this Job.

Results

Once done, a message box pops up to indicate that this file called output.csv does not exist in the directory you defined earlier.

In the HDFS we check the existence of the file, browse to this directory specified, you can see that this file does not exist.
Contents of directory `/user/ychen/data/hdfs/dest`
**tHDFSGet**

Copies files from Hadoop distributed file system (HDFS), pastes them in a user-defined directory and if needs be, renames them.

*tHDFSGet* connects to Hadoop distributed file system, helping to obtain large-scale files with optimized performance.

**tHDFSGet Standard properties**

These properties are used to configure *tHDFSGet* running in the Standard Job framework.

The Standard *tHDFSGet* component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Built-in</em>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><em>Repository</em>: Select the repository file in which the</td>
</tr>
<tr>
<td></td>
<td>properties are stored. The fields that follow are</td>
</tr>
<tr>
<td></td>
<td>completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

| Use an existing connection | Select this check box and in the Component List click   |
|                           | the HDFS connection component from which you want      |
|                           | to reuse the connection details already defined.       |
|                           | Note that when a Job contains the parent Job and the   |
|                           | child Job, Component List presents only the connection  |
|                           | components in the same Job level.                      |

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Select the cluster you are using from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The options in the list vary depending on the component</td>
</tr>
<tr>
<td></td>
<td>you are using. Among these options, the following ones</td>
</tr>
<tr>
<td></td>
<td>requires specific configuration:</td>
</tr>
<tr>
<td></td>
<td>• If available in this Distribution drop-down list, the</td>
</tr>
<tr>
<td></td>
<td>Microsoft HD Insight option allows you to use a Microsoft</td>
</tr>
<tr>
<td></td>
<td>HD Insight cluster. For this purpose, you need to configure</td>
</tr>
<tr>
<td></td>
<td>the connections to the HD Insight cluster and the Windows</td>
</tr>
<tr>
<td></td>
<td>Azure Storage service of that cluster in the areas that</td>
</tr>
<tr>
<td></td>
<td>are displayed. For detailed explanation about these</td>
</tr>
<tr>
<td></td>
<td>parameters, search for configuring the connection manually</td>
</tr>
<tr>
<td></td>
<td>on Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
<tr>
<td></td>
<td>• If you select Amazon EMR, you can find more details</td>
</tr>
<tr>
<td></td>
<td>about how to configure an Amazon EMR cluster in Talend</td>
</tr>
<tr>
<td></td>
<td>• The Custom option allows you to connect to a cluster</td>
</tr>
<tr>
<td></td>
<td>different from any of the distributions given in this list,</td>
</tr>
<tr>
<td></td>
<td>that is to say, to connect to a cluster not</td>
</tr>
<tr>
<td></td>
<td>officially supported by Talend.</td>
</tr>
</tbody>
</table>
1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th>Hadoop version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
</table>

| Use kerberos authentication | If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos. |

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by |
following the explanation in Connecting to a security-enabled MapR on page 1652.

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

This check box is available depending on the Hadoop distribution you are connecting to.

| Use a keytab to authenticate | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is *user1* and the principal to be used is *guest*; in this situation, ensure that *user1* has the right to read the keytab file to be used. |
<p>| NameNode URI | Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called <em>masternode</em> as the NameNode, then the location is hdfs://masternode:portnumber. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a <strong>tLibraryLoad</strong> in the Job to load the library required by the secured WebHDFS. |
| User name | The <strong>User name</strong> field is available when you are not using Kerberos to authenticate. In the <strong>User name</strong> field, enter the login user name for your distribution. If you leave it empty, the user name of the machine hosting the Studio will be used. |
| Group | Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using. |
| HDFS directory | Browse to, or enter the path pointing to the data to be used in the file system. |
| Local directory | Browse to, or enter the local directory to store the files obtained from HDFS. |</p>
<table>
<thead>
<tr>
<th><strong>Overwrite file</strong></th>
<th>Options to overwrite or not the existing file with the new one.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Append</strong></td>
<td>Select this check box to add the new rows at the end of the records.</td>
</tr>
<tr>
<td><strong>Include subdirectories</strong></td>
<td>Select this check box if the selected input source type includes sub-directories.</td>
</tr>
<tr>
<td><strong>Files</strong></td>
<td>In the <strong>Files</strong> area, the fields to be completed are:</td>
</tr>
<tr>
<td></td>
<td>- <strong>File mask</strong>: type in the file name to be selected from HDFS. Regular expression is available.</td>
</tr>
<tr>
<td></td>
<td>- <strong>New name</strong>: give a new name to the obtained file.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>tStatCatcher Statistics</strong></th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hadoop properties</strong></td>
<td><em>Talend Studio</em> uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.</td>
</tr>
<tr>
<td></td>
<td>• Note that if you are using the centrally stored metadata from the <strong>Repository</strong>, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the <strong>Property type</strong> from <strong>Repository</strong> to <strong>Built-in</strong>.</td>
</tr>
<tr>
<td></td>
<td>For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on <a href="http://hadoop.apache.org/docs">http://hadoop.apache.org/docs</a> and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:</td>
</tr>
<tr>
<td></td>
<td>• Typically, the HDFS-related properties can be found in the <code>hdfs-default.xml</code> file of your distribution, such as <a href="http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml">http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml</a>.</td>
</tr>
<tr>
<td></td>
<td>• Apache also provides a page to list the Hive-related properties: <a href="https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties">https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties</a>.</td>
</tr>
</tbody>
</table>
## Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_FILE: the number of files processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CURRENT_STATUS: the execution result of the component. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>TRANSFER_MESSAGES: file transferred information. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

## Usage

### Usage rule

This component combines HDFS connection and data extraction, thus used as a single-component subjob to move data from HDFS to an user-defined local directory.

Different from the tHDFSInput and the tHDFSOutput components, it runs standalone and does not generate input or output flow for the other components.

It is often connected to the Job using OnSubjobOk or OnComponentOk link, depending on the context.

### Dynamic settings

Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of *Talend Studio*.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

Limitations

JRE 1.6+ is required.

Scenario: Computing data with Hadoop distributed file system

This scenario applies only to a Talend solution with Big Data.

The following scenario describes a simple Job that creates a file in a defined directory, get it into and out of HDFS, subsequently store it to another local directory, and read it at the end of the Job.

Setting up the Job

Procedure

1. Drop the following components from the Palette onto the design workspace: tFixedFlowInput, tFileOutputDelimited, tHDFSPut, tHDFSGet, tFileInputDelimited and tLogRow.
2. Connect tFixedFlowInput to tFileOutputDelimited using a Row > Main connection.
3. Connect tFileInputDelimited to tLogRow using a Row > Main connection.
4. Connect tFixedFlowInput to tHDFSPut using an OnSubjobOk connection.
5. Connect `tHDFSPut` to `tHDFSGet` using an `OnSubjobOk` connection.

6. Connect `tHDFSGet` to `tFileInputDelimited` using an `OnSubjobOk` connection.

### Configuring the input component

**Procedure**

1. Double-click `tFixedFlowInput` to define the component in its Basic settings view.

2. Set the **Schema** to **Built-In** and click the three-dot [...] button next to **Edit Schema** to describe the data structure you want to create from internal variables. In this scenario, the schema contains one column: `content`.

3. Click the plus button to add the parameter line.
4. Click **OK** to close the dialog box and accept to propagate the changes when prompted by the studio.

5. In **Basic settings**, define the corresponding value in the **Mode** area using the **Use Single Table** option. In this scenario, the value is "Hello world!".

---

**Configuring the tFileOutputDelimited component**

### Procedure

1. Double-click **tFileOutputDelimited** to define the component in its **Basic settings** view.

   ![tFileOutputDelimited component settings](image)

2. Click the [...] button next to the **File Name** field and browse to the output file you want to write data in, *in.txt* in this example.

---

**Loading the data from the local file**

### Procedure

1. Double-click **tHDFSPut** to define the component in its **Basic settings** view.
2. Select, for example, Apache 0.20.2 from the **Hadoop version** list.

3. In the **NameNode URI**, the **Username** and the **Group** fields, enter the connection parameters to the HDFS. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS.

4. Next to the **Local directory** field, click the three-dot [...] button to browse to the folder with the file to be loaded into the HDFS. In this scenario, the directory has been specified while configuring tFileOutputDelimited: C:/hadoopfiles/putFile/.

5. In the **HDFS directory** field, type in the intended location in HDFS to store the file to be loaded. In this example, it is /testFile.

6. Click the **Overwrite file** field to stretch the drop-down.

7. From the menu, select **always**.

8. In the **Files** area, click the plus button to add a row in which you define the file to be loaded.

9. In the **File mask** column, enter ".*.txt" to replace *newline* between quotation marks and leave the **New name** column as it is. This allows you to extract all the .txt files in the specified directory without changing their names. In this example, the file is *in.txt*.

**Getting the data from the HDFS**

**Procedure**

1. Double-click **tHDFSGet** to define the component in its **Basic settings** view.
2. Select, for example, Apache 0.20.2 from the Hadoop version list.

3. In the NameNode URI, the Username, the Group fields, enter the connection parameters to the HDFS. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS.

4. In the HDFS directory field, type in location storing the loaded file in HDFS. In this example, it is /testFile.

5. Next to the Local directory field, click the three-dot […] button to browse to the folder intended to store the files that are extracted out of the HDFS. In this scenario, the directory is: C:/hadoopfiles/getFile/.

6. Click the Overwrite file field to stretch the drop-down.

7. From the menu, select always.

8. In the Files area, click the plus button to add a row in which you define the file to be extracted.

9. In the File mask column, enter *.txt to replace newLine between quotation marks and leave the New name column as it is. This allows you to extract all the .txt files from the specified directory in the HDFS without changing their names. In this example, the file is in.txt.

Reading data from the HDFS and saving the data locally

Procedure

1. Double-click tFileInputDelimited to define the component in its Basic settings view.
2. Set property type to **Built-In**.

3. Next to the **File Name/Stream** field, click the three-dot button to browse to the file you have obtained from the HDFS. In this scenario, the directory is `C:/hadoopfiles/getFile/in.txt`.

4. Set **Schema** to **Built-In** and click **Edit schema** to define the data to pass on to the tLogRow component.

5. Click the plus button to add a new column.

6. Click **OK** to close the dialog box and accept to propagate the changes when prompted by the studio.

**Executing the Job**

Save the Job and press **F6** to execute it.

The *in.txt* file is created and loaded into the HDFS.
The file is also extracted from the HDFS by `tHDFSGet` and is read by `tFileInputDelimited`.

```
Starting job feature1717_hadoop_get_put at 18:09 04/05/2010.
[statistics] connecting to socket on port 3637
[statistics] connected
hello world!
[statistics] disconnected
Job feature1717_hadoop_get_put ended at 18:09 04/05/2010.
exit code 0
```
**tHDFSInput**

Extracts the data in a HDFS file for other components to process it.

`tHDFSInput` reads a file located on a given Hadoop distributed file system (HDFS) and puts the data of interest from this file into a Talend schema. Then it passes the data to the component that follows.

**tHDFSInput Standard properties**

These properties are used to configure `tHDFSInput` running in the Standard Job framework.

The Standard `tHDFSInput` component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Property type | Either Built-In or Repository.  
|---------------|---------------------------------|
|               | **Built-In**: No property data stored centrally.  
|               | **Repository**: Select the repository file where the properties are stored.  

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.  
|-----------------------|--------------------------------------------------|
|                       | Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
|                       | • **View schema**: choose this option to view the schema only.  
|                       | • **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
|                       | • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.  

| Built-In | You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.  
|----------|--------------------------------------------------|
| Repository | You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.  

Use an existing connection

Select this check box and in the **Component List** click the HDFS connection component from which you want to reuse the connection details already defined.

Note that when a Job contains the parent Job and the child Job, **Component List** presents only the connection components in the same Job level.

Distribution

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this **Distribution** drop-down list, the **Microsoft HD Insight** option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select **Amazon EMR**, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The **Custom** option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by **Talend**.

1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide
range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**
In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see Connecting to a custom Hadoop distribution on page 1589.

<table>
<thead>
<tr>
<th>Hadoop version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
</table>
| **Use kerberos authentication** | If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.

  Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the Force MapR ticket authentication check box and the Use Kerberos authentication check box clear, and then MapR should be able to automatically find that ticket on the fly.

  This check box is available depending on the Hadoop distribution you are connecting to. |
| **Use a keytab to authenticate** | Select the Use a keytab to authenticate check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the Principal field and the access path to the keytab file itself in the Keytab field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

  Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is user1 and the principal to be used is guest; in |
### NameNode URI

Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called *masternode* as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a TLibraryLoad in the Job to load the library required by the secured WebHDFS.

### User name

The **User name** field is available when you are not using Kerberos to authenticate. In the **User name** field, enter the login user name for your distribution. If you leave it empty, the user name of the machine hosting the Studio will be used.

### Group

Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using.

### File Name

Browse to, or enter the path pointing to the data to be used in the file system.

If the path you set points to a folder, this component will read all of the files stored in that folder. Furthermore, if sub-folders exist in that folder and you need to read files in the sub-folders, select the **Include sub-directories if path is directory** check box in the **Advanced settings** view.

### Type

Select the type of the file to be processed. The type of the file may be:

- **Text file**.
- **Sequence file**: a Hadoop sequence file consists of binary key/value pairs and is suitable for the Map/Reduce framework. For further information, see [http://wiki.apache.org/hadoop/SequenceFile](http://wiki.apache.org/hadoop/SequenceFile).

Once you select the **Sequence file** format, the **Key column** list and the **Value column** list appear to allow you to select the keys and the values of that Sequence file to be processed.

### Row separator

The separator used to identify the end of a row. This field is not available for a Sequence file.

### Field separator

Enter character, string or regular expression to separate fields for the transferred data. This field is not available for a Sequence file.

### Header

Set values to ignore the header of the transferred data. For example, enter 0 to ignore no rows for the data without header and set 1 for the data with header at the first row.
<table>
<thead>
<tr>
<th><strong>Custom encoding</strong></th>
<th>You may encounter encoding issues when you process the stored data. In that situation, select this check box to display the <strong>Encoding</strong> list. Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for database data handling. This option is not available for a Sequence file.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compression</strong></td>
<td>Select the <strong>Uncompress the data</strong> check box to uncompress the input data. Hadoop provides different compression formats that help reduce the space needed for storing files and speed up data transfer. When reading a compressed file, the Studio needs to uncompress it before being able to feed it to the input flow. This option is not available for a Sequence file.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Include sub-directories if path is directory</strong></th>
<th>Select this check box to read not only the folder you have specified in the <strong>File name</strong> field but also the sub-folders in that folder.</th>
</tr>
</thead>
</table>
| **Hadoop properties**                           | **Talend Studio** uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.  

- Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.  

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:  

- Typically, the HDFS-related properties can be found in the **hdfs-default.xml** file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).  

- Apache also provides a page to list the Hive-related properties: [https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties). |
### tStatCatcher Statistics
Select this check box to collect log data at the component level. Note that this check box is not available in the Map/Reduce version of the component.

### Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press `Ctrl + Space` to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

**Usage rule**
This component needs an output link.

**Dynamic settings**
Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of *Talend Studio*.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

**Prerequisites**
The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or
libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib
ative. For example, the library for Windows is \lib
ative\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

• Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

| Limitations | JRE 1.6+ is required. |

**Related scenario**

- Related topic, see Scenario 1: Writing data in a delimited file on page 1113.
- Related topic, see Scenario: Computing data with Hadoop distributed file system on page 1513.
tHDFSList

tHDFSList retrieves a list of files or folders based on a filemask pattern and iterates on each unity.

**tHDFSList Standard properties**

These properties are used to configure tHDFSList running in the Standard Job framework.

The Standard tHDFSList component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the HDFS connection component from which you want to reuse the connection details already defined.

Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones require specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add
other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

   In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

   Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

   **Note:**
   In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

   For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th>Hadoop version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use kerberos authentication</strong></td>
<td>If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.</td>
</tr>
<tr>
<td></td>
<td>• If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in <strong>Connecting to a security-enabled MapR</strong> on page 1652.</td>
</tr>
</tbody>
</table>
Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

This check box is available depending on the Hadoop distribution you are connecting to.

| Use a keytab to authenticate | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used. |
| **NameNode URI** | Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called `masternode` as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS. |
| **User name** | The **User name** field is available when you are not using Kerberos to authenticate. In the **User name** field, enter the login user name for your distribution. If you leave it empty, the user name of the machine hosting the Studio will be used. |
| **Group** | Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using. |
| **HDFS Directory** | Browse to, or enter the path pointing to the data to be used in the file system. |
| **FileList Type** | Select the type of input you want to iterate on from the list:

- **Files** if the input is a set of files,
- **Directories** if the input is a set of directories,
- **Both** if the input is a set of the above two types. |
### Include subdirectories
Select this check box if the selected input source type includes sub-directories.

### Case Sensitive
Set the case mode from the list to either create or not create case sensitive filter on filenames.

### Use Glob Expressions as Filemask
This check box is selected by default. It filters the results using a Global Expression (Glob Expressions).

### Files
Click the plus button to add as many filter lines as needed:

- **Filemask**: in the added filter lines, type in a filename or a filemask using special characters or regular expressions.

### Order by
The folders are listed first of all, then the files. You can choose to prioritise the folder and file order either:

- **By default**: alphabetical order, by folder then file;
- **By file name**: alphabetical order or reverse alphabetical order;
- **By file size**: smallest to largest or largest to smallest;
- **By modified date**: most recent to least recent or least recent to most recent.

**Note:**
If ordering by file name, in the event of identical file names then modified date takes precedence. If ordering by file size, in the event of identical file sizes then file name takes precedence. If ordering by modified date, in the event of identical dates then file name takes precedence.

### Order action
Select a sort order by clicking one of the following radio buttons:

- **ASC**: ascending order;
- **DESC**: descending order;

### Advanced settings

#### Use Exclude Filemask
Select this check box to enable Exclude Filemask field to exclude filtering condition based on file type:

- **Exclude Filemask**: Fill in the field with file types to be excluded from the Filemasks in the Basic settings view.

**Note:**
File types in this field should be quoted with double quotation marks and seperated by comma.

#### Hadoop properties
*Talend Studio* uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or...
properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).
- Apache also provides a page to list the Hive-related properties: [https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties).

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</th>
</tr>
</thead>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>CURRENT_FILE</strong>: the current file name. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>CURRENT_FILEDIRECTORY</strong>: the current file directory. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>CURRENT_FILEEXTENSION</strong>: the extension of the current file. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>CURRENT_FILEPATH</strong>: the current file path. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_FILE</strong>: the number of files iterated upon so far. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
For further information about variables, see *Talend Studio User Guide*.

### Usage

#### Usage rule

**tHDFSList** provides a list of files or folders from a defined HDFS directory on which it iterates.

#### Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of *Talend Studio*.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

#### Connections

Outgoing links (from this component to another):

- **Row**: Iterate
- **Trigger**: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error.

Incoming links (from one component to this one):

- **Row**: Iterate.
- **Trigger**: Run if; On Subjob Ok; On Subjob Error; On component Ok; On Component Error; Synchronize; Parallelize.

For further information regarding connections, see *Talend Studio User Guide*.

#### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the
library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

### Limitation

| Limitation | JRE 1.6+ is required. |

### Scenario: Iterating on a HDFS directory

This scenario applies only to a Talend solution with Big Data.

This scenario uses a two-component Job to iterate on a specified directory in HDFS so as to select the files from there towards a local directory.

![Scenario Diagram](image)

### Preparing the data to be used

**Procedure**

Create the files to be iterated on in the HDFS you want to use. In this scenario, two files are created in the directory: `/user/ychen/data/hdfs/out`. 
You can design a Job in the Studio to create the two files. For further information, see `tHDFSPut` on page 1558 or `tHDFSOutput` on page 1538.

### Linking the components

**Procedure**

1. In the **Integration** perspective of **Talend Studio**, create an empty Job, named `HDFSList` for example, from the **Job Designs** node in the **Repository** tree view.
   
   For further information about how to create a Job, see the **Talend Studio User Guide**.

2. Drop `tHDFSList` and `tHDFSGet` onto the workspace.

3. Connect them using the **Row > Iterate** link.

### Configuring the iteration

**Procedure**

1. Double-click `tHDFSList` to open its **Component** view.
2. In the Version area, select the Hadoop distribution you are connecting to and its version.

3. In the Connection area, enter the values of the parameters required to connect to the HDFS.

   In the real-world practice, you may use `tHDFSConnection` to create a connection and reuse it from the current component. For further information, see `tHDFSConnection` on page 1482.

4. In the HDFS Directory field, enter the path to the folder where the files to be iterated on are. In this example, as presented earlier, the directory is `/user/ychen/data/hdfs/out`.

5. In the FileList Type field, select `File`.

6. In the Files table, click to add one row and enter `*` between the quotation marks to iterate on any files existing.

**Selecting the files**

**Procedure**

1. Double-click `tHDFSGet` to open its Component view.
2. In the Version area, select the Hadoop distribution you are connecting to and its version.

3. In the Connection area, enter the values of the parameters required to connect to the HDFS. In the real-world practice, you may have used tHDFSConnection to create a connection; then you can reuse it from the current component. For further information, see tHDFSConnection on page 1482.

4. In the HDFS directory field, enter the path to the folder holding the files to be retrieved. To do this with the auto-completion list, place the mouse pointer in this field, then, press Ctrl + Space to display the list and select the tHDFSLList_1_CURRENT_FILEDIRECTORY variable to reuse the directory you have defined in tHDFSLList. In this variable, tHDFSLList_1 is the label of the component. If you label it differently, select the variable accordingly. Once selecting this variable, the directory reads, for example, 

\[
\text{((String)globalMap.get("tHDFSLList_1_CURRENT_FILEDIRECTORY"))}
\]

in this field. For further information about how to label a component, see the Talend Studio User Guide.

5. In the Local directory field, enter the path, or browse to the folder you want to place the selected files in. This folder will be created if it does not exist. In this example, it is C:/hdfsFiles.

6. In the Overwrite file field, select always.

7. In the Files table, click + to add one row and enter * between the quotation marks in the Filemask column in order to get any files existing.

**Executing the Job**

**Procedure**

Press F6 to execute this Job.
Results

Once done, you can check the files created in the local directory.
tHDFSOoutput

Writes data flows it receives into a given Hadoop distributed file system (HDFS).

**tHDFSOoutput Standard properties**

These properties are used to configure tHDFSOoutput running in the Standard Job framework.

The Standard tHDFSOoutput component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

| Property type | Either Built-In or Repository.  
|---------------|--------------------------------|
|               | **Built-In**: No property data stored centrally.  
|               | **Repository**: Select the repository file where the properties are stored.  

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-In** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Use an existing connection**

Select this check box and in the **Component List** click the HDFS connection component from which you want to reuse the connection details already defined.
Note that when a Job contains the parent Job and the child Job, **Component List** presents only the connection components in the same Job level.

<table>
<thead>
<tr>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:</td>
</tr>
<tr>
<td>• If available in this <strong>Distribution</strong> drop-down list, the <strong>Microsoft HD Insight</strong> option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
<tr>
<td>• If you select <strong>Amazon EMR</strong>, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
<tr>
<td>• The <strong>Custom</strong> option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by <strong>Talend</strong>.</td>
</tr>
<tr>
<td>1. Select <strong>Import from existing version</strong> to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.</td>
</tr>
<tr>
<td>2. Select <strong>Import from zip</strong> to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.</td>
</tr>
<tr>
<td>In <strong>Talend Exchange</strong>, members of <strong>Talend</strong> community have shared some ready-for-use configuration zip files which you can download from this <a href="https://help.talend.com">Hadoop configuration list</a> and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the <strong>Import from existing version</strong> option to take an existing distribution as base to add the jars required by your distribution.</td>
</tr>
<tr>
<td>Note that custom versions are not officially supported by <strong>Talend</strong>. <strong>Talend</strong> and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient</td>
</tr>
</tbody>
</table>
Hadoop experience to handle any issues on your own.

**Note:**
In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see [Connecting to a custom Hadoop distribution](#) on page 1589.

<table>
<thead>
<tr>
<th><strong>Hadoop version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
</table>
| **Use kerberos authentication** | If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.  
- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in [Connecting to a security-enabled MapR](#) on page 1652.  
Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.  
This check box is available depending on the Hadoop distribution you are connecting to. |
| **Use a keytab to authenticate** | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.  
Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used. |
| **NameNode URI** | Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that |
you have chosen a machine called *masternode* as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS.

| **User name** | The **User name** field is available when you are not using Kerberos to authenticate. In the **User name** field, enter the login user name for your distribution. If you leave it empty, the user name of the machine hosting the Studio will be used. |
| **Group** | Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using. |
| **File Name** | Browse to, or enter the location of the file which you write data to. This file is created automatically if it does not exist. |
| **Type** | Select the type of the file to be processed. The type of the file may be:  
  - **Text file**.  
  - **Sequence file**: a Hadoop sequence file consists of binary key/value pairs and is suitable for the Map/Reduce framework. For further information, see [http://wiki.apache.org/hadoop/SequenceFile](http://wiki.apache.org/hadoop/SequenceFile).  
  
  Once you select the **Sequence file** format, the **Key column** list and the **Value column** list appear to allow you to select the keys and the values of that Sequence file to be processed. |
| **Action** | Select an operation in HDFS:  
  - **Create**: Creates a file with data using the file name defined in the **File Name** field.  
  - **Overwrite**: Overwrites the data in the file specified in the **File Name** field.  
  - **Append**: Inserts the data into the file specified in the **File Name** field. The specified file is created automatically if it does not exist. |
| **Row separator** | The separator used to identify the end of a row.  
  This field is not available for a Sequence file. |
| **Field separator** | Enter character, string or regular expression to separate fields for the transferred data.  
  This field is not available for a Sequence file. |
| **Custom encoding** | You may encounter encoding issues when you process the stored data. In that situation, select this check box to display the **Encoding** list. |
Select the encoding from the list or select **Custom** and define it manually. This field is compulsory for database data handling.

This option is not available for a Sequence file.

| Compression      | Select the **Compress the data** check box to compress the output data.  
|                 | Hadoop provides different compression formats that help reduce the space needed for storing files and speed up data transfer. When reading a compressed file, the Studio needs to uncompress it before being able to feed it to the input flow.  
|                 | Note that when the type of the file to be written is **Sequence File**, the compression algorithm is embedded within the container files (the part-files) of this sequence file. These files can be read by a **Talend** component such as **tHDFSInput** within MapReduce Jobs and other applications that understand the sequence file format. Alternatively, when the type is **Text File**, the output files can be accessed with standard compression utilities that understand the **bzip2** or **gzip** container files. |

| Include header  | Select this check box to output the header of the data.  
|                 | This option is not available for a Sequence file. |

### Advanced settings

| Hadoop properties | **Talend Studio** uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.  
|                  | • Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**. |
|                  | For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache's Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:  
|                  | • Typically, the HDFS-related properties can be found in the **hdfs-default.xml** file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).  
|                  | • Apache also provides a page to list the Hive-related properties: [https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties). |
### tStatCatcher Statistics

Select this check box to collect log data at the component level.

### Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

#### Usage rule

This component needs an input component.

#### Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of *Talend Studio*.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489.

For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

#### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each
OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

**Limitations**

<table>
<thead>
<tr>
<th>Limitations</th>
<th>JRE 1.6+ is required.</th>
</tr>
</thead>
</table>

**Related scenario**

- Related topic, see Scenario 1: Writing data in a delimited file on page 1113.
- Related topic, see Scenario: Computing data with Hadoop distributed file system on page 1513.
**tHDFSOutputRaw**

Transfers data of different formats such as hierarchical data in the form of a single column into a given HDFS file system.

tHDFSOutputRaw receives a single-column input flow and writes the data into HDFS.

**tHDFSOutputRaw Standard properties**

These properties are used to configure tHDFSOutputRaw running in the Standard Job framework.

The Standard tHDFSOutputRaw component belongs to the Big Data family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Property type | Either Built-In or Repository.  
|---------------|-------------------------------|
|              | **Built-In**: No property data stored centrally.  
|              | **Repository**: Select the repository file where the properties are stored. |

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

The schema of this component is read-only. You can click **Edit schema** to view the schema.

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Use an existing connection**

Select this check box and in the **Component List** click the HDFS connection component from which you want to reuse the connection details already defined.

Note that when a Job contains the parent Job and the child Job, **Component List** presents only the connection components in the same Job level.

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:
• If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

• If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

• The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the Import from existing version option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

Note:
In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.
For a step-by-step example about how to connect to a custom distribution and share this connection, see [Connecting to a custom Hadoop distribution](#) on page 1589.

<table>
<thead>
<tr>
<th><strong>Hadoop version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use kerberos authentication</strong></td>
<td>If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.</td>
</tr>
<tr>
<td></td>
<td>- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in <a href="#">Connecting to a security-enabled MapR</a> on page 1652.</td>
</tr>
<tr>
<td></td>
<td>Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the <strong>Force MapR ticket authentication</strong> check box and the <strong>Use Kerberos authentication</strong> check box clear, and then MapR should be able to automatically find that ticket on the fly.</td>
</tr>
<tr>
<td></td>
<td>This check box is available depending on the Hadoop distribution you are connecting to.</td>
</tr>
<tr>
<td><strong>Use a keytab to authenticate</strong></td>
<td>Select the <strong>Use a keytab to authenticate</strong> check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the <strong>Principal</strong> field and the access path to the keytab file itself in the <strong>Keytab</strong> field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.</td>
</tr>
<tr>
<td></td>
<td>Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is <em>user1</em> and the principal to be used is <em>guest</em>; in this situation, ensure that <em>user1</em> has the right to read the keytab file to be used.</td>
</tr>
<tr>
<td><strong>NameNode URI</strong></td>
<td>Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called <em>masternode</em> as the NameNode, then the location is <code>hdfs://masternode:portnumber</code>. If you are using WebHDFS, the location should be <code>webhdfs://masternode:portnumber</code>; if this WebHDFS is secured with SSL, the scheme should be <code>swebhdfs</code> and you need to use a <code>tLibraryLoad</code> in the Job to load the library required by the secured WebHDFS.</td>
</tr>
<tr>
<td>Use Datanode hostname</td>
<td>Select the <strong>Use datanode hostname</strong> check box to allow the Job to access datanodes via their hostnames. This actually sets the <code>dfs.client.use.datanode.hostname</code> property to <code>true</code>. When connecting to a S3N filesystem, you must select this check box.</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>User name</td>
<td>The <strong>User name</strong> field is available when you are not using Kerberos to authenticate. In the <strong>User name</strong> field, enter the login user name for your distribution. If you leave it empty, the user name of the machine hosting the Studio will be used.</td>
</tr>
<tr>
<td>Group</td>
<td>Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using.</td>
</tr>
<tr>
<td>File Name</td>
<td>Browse to, or enter the location of the file which you write data to. This file is created automatically if it does not exist.</td>
</tr>
</tbody>
</table>
| Action                | Select an operation in HDFS:  
  * **Create**: Creates a file with data using the file name defined in the **File Name** field.  
  * **Overwrite**: Overwrites the data in the file specified in the **File Name** field.  
  * **Append**: Inserts the data into the file specified in the **File Name** field. The specified file is created automatically if it does not exist. |
| Custom encoding       | You may encounter encoding issues when you process the stored data. In that situation, select this check box to display the **Encoding** list.  
  Select the encoding from the list or select **Custom** and define it manually. This field is compulsory for database data handling.  
  This option is not available for a Sequence file. |
| Compression           | Select the **Compress the data** check box to compress the output data.  
  Hadoop provides different compression formats that help reduce the space needed for storing files and speed up data transfer. When reading a compressed file, the Studio needs to uncompress it before being able to feed it to the input flow.  
  Note that when the type of the file to be written is **Sequence File**, the compression algorithm is embedded within the container files (the part- files) of this sequence file. These files can be read by a **Talend** component such as **tHDFSInput** within MapReduce Jobs and other applications that understand the sequence file format. Alternatively, when the type is **Text File**, the output files can be accessed with standard compression utilities that understand the bzip2 or gzip container files. |
**Die on error**

Select the check box to stop the execution of the Job when an error occurs.

Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a **Row > Reject** link.

**Advanced settings**

**Hadoop properties**

Talend Studio uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).
- Apache also provides a page to list the Hive-related properties: [https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties).

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

**Global Variables**

**FILENAME_PATH**: the path of the input file. This is an After variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

#### Usage rule

This component needs an input component that provides the data of a single column. This column must be labeled to *content* and its type must be **Object**.

For example, you can:

- use **tConvertType** to convert a column from *String* to **Object**, or
- use **tJavaRow** to add the data to be processed into the *globalMap* object so that this data becomes available as a global variable for the other components such as **tFixedFlowInput** to construct this required single column.

For further information about **tConvertType**, see *tConvertType* on page 495.

For further information about **tJavaRow**, see *tJavaRow* on page 1858.

For further information about **tFixedFlowInput**, see *tFixedFlowInput* on page 1202.

For further information about how to use a global variable, see the section describing how to use contexts and variables in *Talend Studio User Guide*.

#### Dynamic settings

Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of *Talend Studio*.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

#### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The
following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

Related Scenario

Once you have properly configured the connection to HDFS for this component, this component works exactly the same way as tFileOutputRaw.

For further information about tFileOutputRaw, see tFileOutputRaw on page 1152.
tHDFSProperties

Creates a single row flow that displays the properties of a file processed in HDFS.

tHDFSProperties Standard properties

These properties are used to configure tHDFSProperties running in the Standard Job framework.

The Standard tHDFSProperties component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

Use an existing connection

Select this check box and in the Component List click the HDFS connection component from which you want to reuse the connection details already defined.

Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.

Distribution

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add
other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th>Hadoop version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Use kerberos authentication</th>
<th>If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in <strong>Connecting to a security-enabled MapR</strong> on page 1652.</td>
</tr>
</tbody>
</table>
Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

This check box is available depending on the Hadoop distribution you are connecting to.

| Use a keytab to authenticate | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used. |
| NameNode URI | Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called `masternode` as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS. |
| User name | The **User name** field is available when you are not using Kerberos to authenticate. In the **User name** field, enter the login user name for your distribution. If you leave it empty, the user name of the machine hosting the Studio will be used. |
| Group | Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using. |
| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

The schema of this component is read-only. You can click **Edit schema** to view the schema. |
| File | Browse to, or enter the path pointing to the data to be used in the file system. |
### Get file checksum

Select this check box to generate and output the MD5 information of the file processed.

Note that this is an HDFS only checksum and not a true MD5 hash that can be compared with the MD5 value obtained, for example, from `tFileInputProperties`. For further information about this component, see `tFileInputProperties` on page 1075.

### Advanced settings

#### Hadoop properties

`Talend Studio` uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the `Repository`, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the `Property type` from `Repository` to `Built-in`.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache's Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).

- Apache also provides a page to list the Hive-related properties: [https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties).

#### tStatCatcher Statistics

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

### Global Variables

#### Global Variables

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the `Die on error` check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th><strong>tHDFSProperties</strong> can be standalone component or send the information it generates to its following component.</th>
</tr>
</thead>
</table>

### Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of *Talend Studio*.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR's documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

  Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the [Preferences] dialog box in
<table>
<thead>
<tr>
<th>Limitation</th>
<th>JRE 1.6+ is required.</th>
</tr>
</thead>
</table>

**Related scenario**

Related topic, see [Procedure](#) on page 1159

Related topic, see [Scenario: Iterating on a HDFS directory](#) on page 1533
tHDFSPut

Connects to Hadoop distributed file system to load large-scale files into it with optimized performance.

tHDFSPut copies files from an user-defined directory, pastes them into a given Hadoop distributed file system (HDFS) and if needs be, renames these files.

**tHDFSPut Standard properties**

These properties are used to configure tHDFSPut running in the Standard Job framework.

The Standard tHDFSPut component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the HDFS connection component from which you want to reuse the connection details already defined.

Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by *Talend*. 
1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see Connecting to a custom Hadoop distribution on page 1589.

<table>
<thead>
<tr>
<th>Hadoop version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
</table>
| Use kerberos authentication | If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.  
- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by |
following the explanation in Connecting to a security-enabled MapR on page 1652.

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

This check box is available depending on the Hadoop distribution you are connecting to.

<table>
<thead>
<tr>
<th>Use a keytab to authenticate</th>
</tr>
</thead>
</table>
| Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is **user1** and the principal to be used is **guest**; in this situation, ensure that **user1** has the right to read the keytab file to be used.

<table>
<thead>
<tr>
<th>NameNode URI</th>
</tr>
</thead>
</table>
| Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called **masternode** as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `LibraryLoad` in the Job to load the library required by the secured WebHDFS.

<table>
<thead>
<tr>
<th>User name</th>
</tr>
</thead>
</table>
| The **User name** field is available when you are not using Kerberos to authenticate. In the **User name** field, enter the login user name for your distribution. If you leave it empty, the user name of the machine hosting the Studio will be used.

<table>
<thead>
<tr>
<th>Group</th>
</tr>
</thead>
</table>
| Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using.

<table>
<thead>
<tr>
<th>Local directory</th>
</tr>
</thead>
</table>
| Local directory where are stored the files to be loaded into HDFS.

<table>
<thead>
<tr>
<th>HDFS directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browse to, or enter the path pointing to the data to be used in the file system.</td>
</tr>
</tbody>
</table>
Overwrite file | Options to overwrite or not the existing file with the new one.

**Use Perl5 Regex Expression as Filemask** | Select this check box if you want to use Perl5 regular expressions in the Files field as file filters. This is useful when the name of the file to be used contains special characters such as parentheses.

For information about Perl5 regular expression syntax, see Perl5 Regular Expression Syntax.

**Files** | In the Files area, the fields to be completed are:

- **File mask**: type in the file name to be selected from the local directory. Regular expression is available.
- **New name**: give a new name to the loaded file.

**Die on error** | This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows.

**Advanced settings**

**tStatCatcher Statistics** | Select this check box to collect log data at the component level.

**Hadoop properties** | Talend Studio uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache's Hadoop documentation on http://hadoop.apache.org/docs and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the hdfs-default.xml file of your distribution, such as http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml.

- Apache also provides a page to list the Hive-related properties: https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties.
Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_FILE: the number of files processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRANSFER_MESSAGES: file transferred information. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component combines HDFS connection and data extraction, thus usually used as a single-component subjob to move data from a user-defined local directory to HDFS.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Different from the tHDFSSInput and the tHDFSOuput components, it runs standalone and does not generate input or output flow for the other components.</td>
</tr>
<tr>
<td></td>
<td>It is often connected to the Job using OnSubjobOk or OnComponentOk link, depending on the context.</td>
</tr>
</tbody>
</table>

Dynamic settings

<table>
<thead>
<tr>
<th>Dynamic settings</th>
<th>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.</td>
</tr>
<tr>
<td></td>
<td>For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.</td>
</tr>
<tr>
<td></td>
<td>For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

**Related scenario**

For related scenario, see Scenario: Computing data with Hadoop distributed file system on page 1513.
**tHDFSRename**

Renames the selected files or specified directory on HDFS.

**tHDFSRename Standard properties**

These properties are used to configure tHDFSRename running in the Standard Job framework.

The Standard tHDFSRename component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add
other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**
In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th>Hadoop version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use kerberos authentication</td>
<td>If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.</td>
</tr>
<tr>
<td></td>
<td>• If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in <strong>Connecting to a security-enabled MapR</strong> on page 1652.</td>
</tr>
</tbody>
</table>
Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

This check box is available depending on the Hadoop distribution you are connecting to.

| **Use a keytab to authenticate** | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is **user1** and the principal to be used is **guest**; in this situation, ensure that **user1** has the right to read the keytab file to be used. |
| **NameNode URI** | Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called **masternode** as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS. |
| **User name** | The **User name** field is available when you are not using Kerberos to authenticate. In the **User name** field, enter the login user name for your distribution. If you leave it empty, the user name of the machine hosting the Studio will be used. |
| **Group** | Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using. |
| **HDFS directory** | Browse to, or enter the path pointing to the data to be used in the file system. |
| **Overwrite file** | Select the options to overwrite or not the existing file with the new one. |
| **Files** | Click the `[+]` button to add the lines you want to use as filters: |
### tHDFSRename

<table>
<thead>
<tr>
<th><strong>Filemask</strong></th>
<th>enter the filename or filemask using wildcharacters (*) or regular expressions.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>New name</strong></td>
<td>name to give to the HDFS file after the transfer.</td>
</tr>
</tbody>
</table>

| **Die on error** | This check box is selected by default. Clear the check box to skip the row in error and complete the process for error-free rows. |

### Advanced settings

<table>
<thead>
<tr>
<th><strong>tStatCatcher Statistics</strong></th>
<th>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hadoop properties</strong></td>
<td>Talend Studio uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.</td>
</tr>
<tr>
<td></td>
<td>• Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.</td>
</tr>
<tr>
<td></td>
<td>For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on <a href="http://hadoop.apache.org/docs">http://hadoop.apache.org/docs</a> and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:</td>
</tr>
<tr>
<td></td>
<td>• Typically, the HDFS-related properties can be found in the hdfs-default.xml file of your distribution, such as <a href="http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml">http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml</a>.</td>
</tr>
<tr>
<td></td>
<td>• Apache also provides a page to list the Hive-related properties: <a href="https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties">https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties</a>.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>NB_FILE</strong>: the number of files processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>CURRENT_STATUS</strong>: the execution result of the component. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions</td>
</tr>
</tbody>
</table>
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used to compose a single-component Job or Subjob.</th>
</tr>
</thead>
</table>

#### Dynamic settings

Click the [*] button to add a row in the table and fill the **Code** field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of *Talend Studio*.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

#### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).
Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the `-Djava.library.path` argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the [Window] menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

**Limitation**

| JRE 1.6+ is required. |

**Related scenario**

For related scenario, see Scenario: Computing data with Hadoop distributed file system on page 1513.
tHDFSRowCount

Reads a file in HDFS row by row in order to determine the number of rows this file contains.

tHDFSRowCount counts the number of rows in a file in HDFS. If the file to be processed is a Hadoop sequence file type or a large dataset, it is recommended to use a tAggregateRow to count the records.

**tHDFSRowCount Standard properties**

These properties are used to configure tHDFSRowCount running in the Standard Job framework.

The Standard tHDFSRowCount component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td></td>
</tr>
<tr>
<td>Built-In:</td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Repository:</td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the HDFS connection component from which you want to reuse the connection details already defined.

Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.

**Distribution**

Select the cluster you are using from the drop-down list.

The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.
1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jars which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this [Hadoop configuration list](#) and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th>Hadoop version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use kerberos authentication</td>
<td>If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.</td>
</tr>
<tr>
<td></td>
<td>• If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by</td>
</tr>
</tbody>
</table>
following the explanation in Connecting to a security-enabled MapR on page 1652.

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

This check box is available depending on the Hadoop distribution you are connecting to.

| **Use a keytab to authenticate** | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver. Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is **user1** and the principal to be used is **guest**; in this situation, ensure that **user1** has the right to read the keytab file to be used. |
| **NameNode URI** | Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called **masternode** as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS. |
| **User name** | The **User name** field is available when you are not using Kerberos to authenticate. In the **User name** field, enter the login user name for your distribution. If you leave it empty, the user name of the machine hosting the Studio will be used. |
| **Group** | Enter the membership including the authentication user under which the HDFS instances were started. This field is available depending on the distribution you are using. |
| **File name** | Browse to, or enter the path pointing to the data to be used in the file system. |
| **Row separator** | The separator used to identify the end of a row. |
| **Ignore empty rows** | Select this check box to skip the empty rows. |
### tHDFSRowCount

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for database data handling.</th>
</tr>
</thead>
</table>
| Compression | Select the **Uncompress the data** check box to uncompress the input data.  
Hadoop provides different compression formats that help reduce the space needed for storing files and speed up data transfer. When reading a compressed file, the Studio needs to uncompress it before being able to feed it to the input flow. |

### Advanced settings

**Hadoop properties**

Talend Studio uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache's Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).

- Apache also provides a page to list the Hive-related properties: [https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties).

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

### Global Variables

**Global Variables**

- **COUNT**: the number of rows in a file. This is a Flow variable and it returns an integer.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

#### Usage rule

**tHDFSRowCount** is a standalone component; it must be used with a **OnSubjobOk** connection to **tJava** in order to return the row count.

The valid code for **tJava** to get this count could be:

```java
System.out.print(((Integer)globalMap.get("tHDFSRowCount_1_COUNT")));
```

In this example, **tHDFSRowCount_1** is the label of this component in a Job, so it may vary among different use cases; **COUNT** is the global variable of **tHDFSRowCount**, representing the integer flow of the row count.

For further information about how to label a component or how to use a global variable in a Job, see the *Talend Studio User Guide*.

#### Dynamic settings

Click the [*+] button to add a row in the table and fill the **Code** field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of *Talend Studio*.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.
Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

Limitation

JRE 1.6+ is required.

Related scenarios

No scenario is available for the Standard version of this component yet.
tHiveClose

Closes connection to a Hive database.
tHiveClose closes an active connection to a database.

**tHiveClose Standard properties**

These properties are used to configure tHiveClose running in the Standard Job framework.
The Standard tHiveClose component belongs to the Big Data and the Databases families.
The component in this framework is generally available.

**Basic settings**

| Component list | If there is more than one connection used in the Job, select **tHiveConnection** from the list. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect the log data at a component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see [Talend Studio User Guide](#). |

**Usage**

| Usage rule | This component is to be used along with other Hive components, especially with **tHiveConnection** as **tHiveConnection** allows you to open a connection for the transaction which is underway. |

| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having |
the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Prerequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.</td>
</tr>
</tbody>
</table>

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR's documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tHiveConnection

Establishes a Hive connection to be reused by other Hive components in your Job.
tHiveConnection opens a connection to a Hive database.

**tHiveConnection Standard properties**

These properties are used to configure tHiveConnection running in the Standard Job framework.
The Standard tHiveConnection component belongs to the Big Data, the Databases and the ELT families.
The component in this framework is generally available.

**Basic settings**

Connection configuration:
- When you use this component with Google Dataproc:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project identifier</strong></td>
<td>Enter the ID of your Google Cloud Platform project. If you are not certain about your project ID, check it in the Manage Resources page of your Google Cloud Platform services.</td>
</tr>
<tr>
<td><strong>Cluster identifier</strong></td>
<td>Enter the ID of your Dataproc cluster to be used.</td>
</tr>
<tr>
<td><strong>Region</strong></td>
<td>Enter the geographic zones in which the computing resources are used and your data is stored and processed. If you do not need to specify a particular region, leave the default value global. For further information about the available regions and the zones each region groups, see Regions and Zones.</td>
</tr>
<tr>
<td><strong>Google Storage staging bucket</strong></td>
<td>As a Talend Job expects its dependent jar files for execution, specify the Google Storage directory to which these jar files are transferred so that your Job can access these files at execution. The directory to be entered must end with a slash (/). If not existing, the directory is created on the fly but the bucket to be used must already exist.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Fill this field with the name of the database.</td>
</tr>
<tr>
<td><strong>Provide Google Credentials in file</strong></td>
<td>Leave this check box clear, when you launch your Job from a given machine in which Google Cloud SDK has been installed and authorized to use your user account credentials to access Google Cloud Platform. In this situation, this machine is often your local machine. For further information about this Google Credentials file, see the administrator of your Google Cloud Platform or visit Google Cloud Platform Auth Guide.</td>
</tr>
</tbody>
</table>
When you use this component with HDInsight:

<table>
<thead>
<tr>
<th>WebHCat configuration</th>
<th>Enter the address and the authentication information of the Microsoft HD Insight cluster to be used. For example, the address could be your_hdinsight_cluster_name.azurehdinsight.net and the authentication information is your Azure account name: ychen. The Studio uses this service to submit the Job to the HD Insight cluster. In the Job result folder field, enter the location in which you want to store the execution result of a Job in the Azure Storage to be used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDInsight configuration</td>
<td>Enter the authentication information of the HD Insight cluster to be used.</td>
</tr>
<tr>
<td>Windows Azure Storage configuration</td>
<td>Enter the address and the authentication information of the Azure Storage account to be used. In this configuration, you do not define where to read or write your business data but define where to deploy your Job only. Therefore always use the Azure Blob Storage system for this configuration. In the Container field, enter the name of the container to be used. In the Deployment Blob field, enter the location in which you want to store the current Job and its dependent libraries in this Azure Storage account.</td>
</tr>
<tr>
<td>Database</td>
<td>Fill this field with the name of the database.</td>
</tr>
</tbody>
</table>

When you use the other distributions:

<table>
<thead>
<tr>
<th>Connection mode</th>
<th>Select a connection mode from the list. The options vary depending on the distribution you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hive server</td>
<td>Select the Hive server through which you want the Job using this component to execute queries on Hive. This Hive server list is available only when the Hadoop distribution to be used such as HortonWorks Data Platform V1.2.0 (Bimota) supports HiveServer2. It allows you to select HiveServer2 (Hive 2), the server that better support concurrent connections of multiple clients than HiveServer (Hive 1). For further information about HiveServer2, see <a href="https://cwiki.apache.org/confluence/display/Hive/Setting+Up+HiveServer2">https://cwiki.apache.org/confluence/display/Hive/Setting+Up+HiveServer2</a>.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Fill this field with the name of the database.</td>
</tr>
</tbody>
</table>

Note:
This field is not available when you select Embedded from the Connection mode list.

<table>
<thead>
<tr>
<th><strong>Username and Password</strong></th>
<th>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</th>
</tr>
</thead>
</table>

| **Use kerberos authentication** | If you are accessing a Hive Metastore running with Kerberos security, select this check box and then enter the relevant parameters in the fields that appear.  

  - If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.  

  Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the Force MapR ticket authentication check box and the Use Kerberos authentication check box clear, and then MapR should be able to automatically find that ticket on the fly.  

  The values of the following parameters can be found in the hive-site.xml file of the Hive system to be used.  

  1. Hive principal uses the value of hive.metastore.kerberos.principal. This is the service principal of the Hive Metastore.  

  2. HiveServer2 local user principal uses the value of hive.server2.authentication.kerberos.principal.  

  3. HiveServer2 local user keytab uses the value of hive.server2.authentication.kerberos.keytab  

  4. Metastore URL uses the value of javax.jdo.option.ConnectionURL. This is the JDBC connection string to the Hive Metastore.  

  5. Driver class uses the value of javax.jdo.option.ConnectionDriverName. This is the name of the driver for the JDBC connection.  

  6. Username uses the value of javax.jdo.option.ConnectionUserName. This, as well as the Password parameter, is the user credential for connecting to the Hive Metastore.  

  7. Password uses the value of javax.jdo.option.ConnectionPassword.  

  For the other parameters that are displayed, please consult the Hadoop configuration files they belong to. For example, the Namenode principal can be found in the hdfs-site.xml file or the hdfs-default.xml file of the distribution you are using. |
| **Use a keytab to authenticate** | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is *user1* and the principal to be used is *guest*; in this situation, ensure that *user1* has the right to read the keytab file to be used. |
| **Use SSL encryption** | Select this check box to enable the SSL or TLS encrypted connection.

Then in the fields that are displayed, provide the authentication information:

- In the **Trust store path** field, enter the path, or browse to the TrustStore file to be used. By default, the supported TrustStore types are **JKS** and **PKCS 12**.
- To enter the password, click the [ ... ] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

This feature is available only to the HiveServer2 in the **Standalone** mode of the following distributions:

- Hortonworks Data Platform 2.0 +
- Cloudera CDH4 +
- Pivotal HD 2.0 +
- Amazon EMR 4.0.0 + |
| **Set Resource Manager** | Select this check box and in the displayed field, enter the location of the ResourceManager of your distribution. For example, *tal-qall4.tale nd.lan:8050*.

Then you can continue to set the following parameters depending on the configuration of the Hadoop cluster to be used (if you leave the check box of a parameter clear, then at runtime, the configuration about this parameter in the Hadoop cluster to be used will be ignored):

1. Select the **Set resourcemanager scheduler address** check box and enter the Scheduler address in the field that appears. |
2. Select the **Set jobhistory address** check box and enter the location of the JobHistory server of the Hadoop cluster to be used. This allows the metrics information of the current Job to be stored in that JobHistory server.

3. Select the **Set staging directory** check box and enter this directory defined in your Hadoop cluster for temporary files created by running programs. Typically, this directory can be found under the `yarn.app.mapreduce.am.staging-dir` property in the configuration files such as `yarn-site.xml` or `mapred-site.xml` of your distribution.

4. Allocate proper memory volumes to the **Map** and the **Reduce** computations and the **ApplicationMaster** of YARN by selecting the **Set memory** check box in the **Advanced settings** view.

5. Select the **Set Hadoop user** check box and enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed.

6. Select the **Use datanode hostname** check box to allow the Job to access datanodes via their hostnames. This actually sets the `dfs.client.use.datanode.hostname` property to `true`. When connecting to a S3N filesystem, you must select this check box.

For further information about these parameters, see the documentation or contact the administrator of the Hadoop cluster to be used.


<table>
<thead>
<tr>
<th>Set NameNode URI</th>
</tr>
</thead>
</table>
| Select this check box and in the displayed field, enter the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, assuming that you have chosen a machine called `masternode` as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS.


The other properties:
| Property type | Either Built-In or Repository.  
|---------------|--------------------------------------------------|
| | **Built-In**: No property data stored centrally.  
| | **Repository**: Select the repository file where the properties are stored.  

| Distribution | Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:  
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | • If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).  
| | • If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).  
| | • The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.  
| | 1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.  
| | 2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.  
| | In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.  
| | Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions.
that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**
In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see [Connecting to a custom Hadoop distribution](#) on page 1589.

<table>
<thead>
<tr>
<th>Hive version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
</table>

**Inspect the classpath for configurations**

Select this check box to allow the component to check the configuration files in the directory you have set with the `$HADOOP_CONF_DIR` variable and directly read parameters from these files in this directory. This feature allows you to easily change the Hadoop configuration for the component to switch between different environments, for example, from a test environment to a production environment.

In this situation, the fields or options used to configure Hadoop connection and/or Kerberos security are hidden. If you want to use certain parameters such as the Kerberos parameters but these parameters are not included in these Hadoop configuration files, you need to create a file called `talend-site.xml` and put this file into the same directory defined with `$HADOOP_CONF_DIR`. This `talend-site.xml` file should read as follows:

```xml
<!-- Put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>talend.kerberos.authentication</name>
    <value>kinit</value>
    <description>Set the Kerberos authentication method to use. Valid values are: kinit or keytab. </description>
  </property>
  <property>
    <name>talend.kerberos.keytab.principal</name>
    <value>user@BIGDATA.COM</value>
    <description>Set the keytab's principal name. </description>
  </property>
</configuration>
```
The parameters read from these configuration files override the default ones used by the Studio. When a parameter does not exist in these configuration files, the default one is used.

Note that this option is available only in Hive Standalone mode with Hive 2.

| Use or register a shared DB Connection | Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the Shared DB Connection Name field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the Use dynamic job and Use an independent process to run subjob options of the tRunJob component. Using a shared connection together with a tRunJob component with either of these two options enabled will cause your Job to fail. |
| Execution engine | Select this check box and from the drop-down list, select the framework you need to use to run the Job.

This list is available only when you are using the Embedded mode for the Hive connection and the |
Before using Tez, ensure that the Hadoop cluster you are using supports Tez. You will need to configure the access to the relevant Tez libraries via the **Advanced settings** view of this component.

For further information about Hive on Tez, see Apache’s related documentation in https://cwiki.apache.org/confluence/display/Hive/Hive+on+Tez. Some examples are presented there to show how Tez can be used to gain performance over MapReduce.

---

**Store by HBase**

Select this check box to display the parameters to be set to allow the Hive components to access HBase tables:

- Once this access is configured, you will be able to use, in `tHiveRow` and `tHiveInput`, the Hive QL statements to read and write data in HBase.
- If you are using the Kerberos authentication, you need to define the HBase related principals in the corresponding fields that are displayed.

For further information about this access involving Hive and HBase, see Apache’s Hive documentation about Hive/HBase integration.

**Zookeeper quorum**

Type in the name or the URL of the Zookeeper service you use to coordinate the transaction between your Studio and your database. Note that when you configure the Zookeeper, you might need to explicitly set the `zookeeper.znode.parent` property to define the path to the root znode that contains all the znodes created and used by your database; then select the **Set Zookeeper znode parent** check box to define this property.

**Zookeeper client port**

Type in the number of the client listening port of the Zookeeper service you are using.

**Define the jars to register for HBase**

Select this check box to display the **Register jar for HBase** table, in which you can register any missing jar file required by HBase, for example, the Hive Storage Handler, by default, registered along with your Hive installation.

**Register jar for HBase**

Click the `[+]` button to add rows to this table, then, in the **Jar name** column, select the jar file(s) to be registered and in the **Jar path** column, enter the path(s) pointing to that or those jar file(s).
### Advanced settings

| **Tez lib** | Select how the Tez libraries are accessed:  

- **Auto install**: at runtime, the Job uploads and deploys the Tez libraries provided by the Studio into the directory you specified in the **Install folder in HDFS** field, for example, `/tmp/usr/tez`.  
  
  If you have set the `tez.lib.uris` property in the properties table, this directory overrides the value of that property at runtime. But the other properties set in the properties table are still effective.  

- **Use exist**: the Job accesses the Tez libraries already deployed in the Hadoop cluster to be used. You need to enter the path pointing to those libraries in the **Lib path (folder or file)** field.  

- **Lib jar**: this table appears when you have selected **Auto install** from the **Tez lib** list and the distribution you are using is **Custom**. In this table, you need to add the Tez libraries to be uploaded. |

| **Hadoop properties** | *Talend Studio* uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.  

- Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.  

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache's Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:  

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).  

- Apache also provides a page to list the Hive-related properties: [https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties). |

| **Hive properties** | *Talend Studio* uses a default configuration for its engine to perform operations in a Hive database. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones. For further information for Hive dedicated properties, see [https://](https://) |
• If you need to use Tez to run your Hive Job, add `hive.execution.engine` to the Properties column and `Tez` to the Value column, enclosing both of these strings in double quotation marks.

• Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.

**Mapred job map memory mb** and **Mapred job reduce memory mb**
You can tune the map and reduce computations by selecting the Set memory check box to set proper memory allocations for the computations to be performed by the Hadoop system.

In that situation, you need to enter the values you need in the **Mapred job map memory mb** and the **Mapred job reduce memory mb** fields, respectively. By default, the values are both 1000 which are normally appropriate for running the computations.

**Path separator in server**
Leave the default value of the Path separator in server as it is, unless you have changed the separator used by your Hadoop distribution’s host machine for its PATH variable or in other words, that separator is not a colon (:). In that situation, you must change this value to the one you are using in that host.

**tStatCatcher Statistics**
Select this check box to collect the log data at a component level.

### Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](https://cwiki.apache.org/confluence/display/Hive/AdminManual+Configuration).

### Usage

**Usage rule**
This component is generally used with other Hive components, particularly **tHiveClose**.

If the Studio used to connect to a Hive database is operated on Windows, you must manually create a folder
<table>
<thead>
<tr>
<th>HiveConnection</th>
<th>called tmp in the root of the disk where this Studio is installed.</th>
</tr>
</thead>
</table>

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the [Preferences] dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the **Data viewer** to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

### Connecting to a custom Hadoop distribution

As explained in the properties table, when you select the **Custom** option from the **Distribution** drop-down list, you are connecting to a Hadoop distribution different from any of the Hadoop distributions provided on that **Distribution** list in the Studio.

After selecting this **Custom** option, click the button to display the [Import custom definition] dialog box and proceed as follows:

**Procedure**

1. Depending on your situation, select **Import from existing version** or **Import from zip** to configure the custom Hadoop distribution to be connected to.
   - If you have the zip file of the custom Hadoop distribution you need to connect to, select **Import from zip**. *Talend* community provides this kind of zip files that you can download from [http://www.talendforge.org/exchange/index.php](http://www.talendforge.org/exchange/index.php).
   - Otherwise, select **Import from existing version** to import an officially supported Hadoop distribution as base so as to customize it by following the wizard.
Note that the check boxes in the wizard allow you to select the Hadoop element(s) you need to import. All the check boxes are not always displayed in your wizard depending on the context in which you are creating the connection. For example, if you are creating this connection for a Hive component, then only the Hive check box appears.

2. Whether you have selected **Import from existing version** or **Import from zip**, verify that each check box next to the Hadoop element you need to import has been selected.

3. Click **OK** and then in the pop-up warning, click **Yes** to accept overwriting any custom setup of jar files previously implemented.

Once done, the [Custom Hadoop version definition] dialog box becomes active.
4. If you have selected **Import from zip**, click **OK** to validate the imported configuration.

If you have selected **Import from existing version** as base, you should still need to add more jar files to customize that version. Then from the tab of the Hadoop element you need to customize, for example, the **HDFS/HCatalog/Oozie** tab, click the **[+]** button to open the **[Select libraries]** dialog box.

5. Select the **External libraries** option to open its view.

6. Browse to and select any jar file you need to import.

7. Click **OK** to validate the changes and to close the **[Select libraries]** dialog box.

   Once done, the selected jar file appears on the list in the tab of the Hadoop element being configured.

   Note that if you need to share the custom Hadoop setup with another Studio, you can export this custom connection from the **[Custom Hadoop version definition]** window using the **[button.**

8. In the **[Custom Hadoop version definition]** dialog box, click **OK** to validate the customized configuration. This brings you back to the **Distribution list in the Basic settings** view of the component.

**Results**

Now that the configuration of the custom Hadoop version has been set up and you are back to the **Distribution list**, you are able to continue to enter other parameters required by the connection.
If the custom Hadoop version you need to connect to contains YARN and you want to use it, select the **Use YARN** check box next to the **Distribution** list.

A video is available in the following link to demonstrate, by taking HDFS as example, how to set up the connection to a custom Hadoop cluster, also referred to as an unsupported Hadoop distribution: **How to add an unsupported Hadoop distribution to the Studio.**

**Scenario: creating a partitioned Hive table**

This scenario illustrates how to use **tHiveConnection**, **tHiveCreateTable** and **tHiveLoad** to create a partitioned Hive table and write data in it.

Note that **tHiveCreateTable** and **tHiveLoad** are available only when you are using one of the **Talend** solutions with Big Data.

The sample data to be used in this scenario is employee information of a company, reading as follows:

<table>
<thead>
<tr>
<th></th>
<th>First Name</th>
<th>Last Name</th>
<th>Date of Birth</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lyndon</td>
<td>Fillmore</td>
<td>21-05-2008</td>
<td>US</td>
</tr>
<tr>
<td>2</td>
<td>Ronald</td>
<td>McKinley</td>
<td>15-08-2008</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Ulysses</td>
<td>Roosevelt</td>
<td>05-10-2008</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Harry</td>
<td>Harrison</td>
<td>23-11-2007</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Lyndon</td>
<td>Garfield</td>
<td>19-07-2007</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>James</td>
<td>Quincy</td>
<td>15-07-2008</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Chester</td>
<td>Jackson</td>
<td>26-02-2008</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Dwight</td>
<td>McKinley</td>
<td>16-07-2008</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Jimmy</td>
<td>Johnson</td>
<td>23-12-2007</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Herbert</td>
<td>Fillmore</td>
<td>03-04-2008</td>
<td></td>
</tr>
</tbody>
</table>

The information contains some employees' names and the dates when they are registered in a HR system. Since these employees work for the US subsidiary of the company, you will create a US partition for this sample data.

Before starting to replicate this scenario, ensure that you have appropriate rights and permissions to access the Hive database to be used.

Note that if you are using the Windows operating system, you have to create a **tmp** folder at the root of the disk where the Studio is installed.
Then proceed as follows:

**Linking the components**

**Procedure**

1. In the **Integration** perspective of the Studio, create an empty Job from the **Job Designs** node in the **Repository** tree view.
   
   For further information about how to create a Job, see the chapter describing how to designing a Job in *Talend Studio User Guide*.

2. Drop **tHiveConnection**, **tHiveCreateTable** and **tHiveLoad** onto the workspace.

3. Connect them using the **Trigger > On Subjob OK** link.

**Configuring the connection to Hive**

**About this task**

Configuring **tHiveConnection**

**Procedure**

1. Double-click **tHiveConnection** to open its **Component** view.

2. From the **Property type** list, select **Built-in**. If you have created the connection to be used in **Repository**, then select **Repository**, click the **...** button to open the [**Repository content**](#) dialog box and select that connection. This way, the Studio will reuse that set of connection information for this Job.
For further information about how to create a Hadoop connection in Repository, see the chapter describing the Hadoop cluster node of the Talend Open Studio for Big Data Getting Started Guide.

3. In the Version area, select the Hadoop distribution to be used and its version. If you cannot find from the list the distribution corresponding to yours, select Custom so as to connect to a Hadoop distribution not officially supported in the Studio.

For a step-by-step example about how to use this Custom option, see Connecting to a custom Hadoop distribution on page 1589.

4. In the Connection area, enter the connection parameters to the Hive database to be used.

5. In the Name node field, enter the location of the master node, the NameNode, of the distribution to be used. For example, talend-hdp-all:50300. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS.

6. In the Job tracker field, enter the location of the JobTracker of your distribution. For example, hdfs://talend-hdp-all:8020.

Note that the notion Job in this term JobTracker designates the MR or the MapReduce jobs described in Apache’s documentation on http://hadoop.apache.org/.

Configuring tHiveConnection

Procedure

1. Double-click tHiveConnection to open its Component view.

2. From the Property type list, select Built-in. If you have created the connection to be used in Repository, then select Repository, click the button to open the [Repository content] dialog.
box and select that connection. This way, the Studio will reuse that set of connection information for this Job.

For further information about how to create a Hadoop connection in Repository, see the chapter describing the Hadoop cluster node of the Talend Open Studio for Big Data Getting Started Guide.

3. In the Version area, select the Hadoop distribution to be used and its version. If you cannot find from the list the distribution corresponding to yours, select Custom so as to connect to a Hadoop distribution not officially supported in the Studio.

For a step-by-step example about how to use this Custom option, see Connecting to a custom Hadoop distribution on page 1589.

4. In the Connection area, enter the connection parameters to the Hive database to be used.

5. In the Name node field, enter the location of the master node, the NameNode, of the distribution to be used. For example, talend-hdp-all:50300. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS.

6. In the Job tracker field, enter the location of the JobTracker of your distribution. For example, hdfs://talend-hdp-all:8020.

Note that the notion Job in this term JobTracker designates the MR or the MapReduce jobs described in Apache's documentation on http://hadoop.apache.org/.

Creating the Hive table

Defining the schema

Procedure

1. Double-click tHiveCreateTable to open its Component view.

2. Select the Use an existing connection check box and from Component list, select the connection configured in the tHiveConnection component you are using for this Job.
3. Click the button next to **Edit schema** to open the schema editor.

4. Click the button four times to add four rows and in the **Column** column, rename them to *Id*, *FirstName*, *LastName* and *Reg_date*, respectively.

5. In the **Type** column, select the type of the data in each column. In this scenario, *Id* is of the **Integer** type, *Reg_date* is of the **Date** type and the others are of the **String** type.

6. In the **DB type** column, select the Hive type of each column corresponding to their data types you have defined. For example, *Id* is of **INT** and *Reg_date* is of **TIMESTAMP**.

7. In the **Data pattern** column, define the pattern corresponding to that of the raw data. In this example, use the default one.

8. Click **OK** to validate these changes.

**Defining the table settings**

**Procedure**

1. In **Table name** field, enter the name of the Hive table to be created. In this scenario, it is *employees*.
2. From the **Action on table** list, select **Create table if not exists**.
3. From the **Format** list, select the data format that this Hive table in question is created for. In this scenario, it is **TEXTFILE**.
4. Select the **Set partitions** check box to add the *US* partition as explained at the beginning of this scenario. To define this partition, click the button next to **Edit schema** that appears.
5. Leave the **Set file location** check box clear to use the default path for Hive table.
6. Select the **Set Delimited row format** check box to display the available options of row format.
7. Select the **Field** check box and enter a semicolon (,) as field separator in the field that appears.
8. Select the **Line** check box and leave the default value as line separator.
Writing data to the table

About this task
Configuring tHiveLoad

Procedure

1. Double-click **tHiveLoad** to open its **Component** view.

2. Select the **Use an existing connection** check box and from **Component list**, select the connection configured in the **tHiveConnection** component you are using for this Job.

3. From the **Load action** field, select **LOAD** to write data from the file holding the sample data that is presented at the beginning of this scenario.

4. In the **File path** field, enter the directory where the sample data is stored. In this example, the data is stored in the HDFS system to be used.

   In the real-world practice, you can use **tHDFSOutput** to write data into the HDFS system and you need to ensure that the Hive application has the appropriate rights and permissions to read or even move the data.

   For further information about **tHDFSOutput**, see **tHDFSOutput** on page 1538.

   for further information about the related rights and permissions, see the documentation or contact the administrator of the Hadoop cluster to be used.

   Note if you need to read data from a local file system other than the HDFS system, ensure that the data to be read is stored in the local file system of the machine in which the Job is run and then select the **Local** check box in this **Basic settings** view. For example, when the connection mode to Hive is **Standalone**, the Job is run in the machine where the Hive application is installed and thus the data should be stored in that machine.

5. In the **Table name** field, enter the name of the target table you need to load data in. In this scenario, it is **employees**.

6. From the **Action on file** list, select **APPEND**.

7. Select the **Set partitions** check box and in the field that appears, enter the partition you need to add data to. In this scenario, this partition is **country='US'**.
Configuring tHiveLoad

Procedure

1. Double-click tHiveLoad to open its Component view.

![tHiveLoad Component view](image)

2. Select the Use an existing connection check box and from Component list, select the connection configured in the tHiveConnection component you are using for this Job.

3. From the Load action field, select LOAD to write data from the file holding the sample data that is presented at the beginning of this scenario.

4. In the File path field, enter the directory where the sample data is stored. In this example, the data is stored in the HDFS system to be used. In the real-world practice, you can use tHDFSOutput to write data into the HDFS system and you need to ensure that the Hive application has the appropriate rights and permissions to read or even move the data.

   For further information about the related rights and permissions, see the documentation or contact the administrator of the Hadoop cluster to be used.

   Note if you need to read data from a local file system other than the HDFS system, ensure that the data to be read is stored in the local file system of the machine in which the Job is run and then select the Local check box in this Basic settings view. For example, when the connection mode to Hive is Standalone, the Job is run in the machine where the Hive application is installed and thus the data should be stored in that machine.

5. In the Table name field, enter the name of the target table you need to load data in. In this scenario, it is employees.

6. From the Action on file list, select APPEND.

7. Select the Set partitions check box and in the field that appears, enter the partition you need to add data to. In this scenario, this partition is country='US'.

Executing the Job

Then you can press F6 to run this Job.

Once done, the Run view is opened automatically, where you can check the execution process.

You can as well verify the results in the web console of the Hadoop distribution used.
If you need to obtain more details about the Job, it is recommended to use the web console of the Jobtracker provided by the Hadoop distribution you are using.
**tHiveCreateTable**

Creates Hive tables that fit a wide range of Hive data formats.

A proper Hive data format such as RC or ORC allows you to obtain a better performance in processing data with Hive.

tHiveCreateTable connects to the Hive database to be used and creates a Hive table that is dedicated to data of the format you specify.

**tHiveCreateTable Standard properties**

These properties are used to configure tHiveCreateTable running in the Standard Job framework.

The Standard tHiveCreateTable component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

Connection configuration:

- When you use this component with Google Dataproc:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project identifier</strong></td>
<td>Enter the ID of your Google Cloud Platform project. If you are not certain about your project ID, check it in the Manage Resources page of your Google Cloud Platform services.</td>
</tr>
<tr>
<td><strong>Cluster identifier</strong></td>
<td>Enter the ID of your Dataproc cluster to be used.</td>
</tr>
<tr>
<td><strong>Region</strong></td>
<td>Enter the geographic zones in which the computing resources are used and your data is stored and processed. If you do not need to specify a particular region, leave the default value <code>global</code>. For further information about the available regions and the zones each region groups, see Regions and Zones.</td>
</tr>
<tr>
<td><strong>Google Storage staging bucket</strong></td>
<td>As a Talend Job expects its dependent jar files for execution, specify the Google Storage directory to which these jar files are transferred so that your Job can access these files at execution. The directory to be entered must end with a slash (/). If not existing, the directory is created on the fly but the bucket to be used must already exist.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Fill this field with the name of the database.</td>
</tr>
<tr>
<td><strong>Provide Google Credentials in file</strong></td>
<td>Leave this check box clear, when you launch your Job from a given machine in which Google Cloud SDK has been installed and authorized to use your user account credentials to access Google Cloud Platform. In this situation, this machine is often your local machine.</td>
</tr>
</tbody>
</table>
For further information about this Google Credentials file, see the administrator of your Google Cloud Platform or visit Google Cloud Platform Auth Guide.

- When you use this component with HDInsight:

| WebHCat configuration | Enter the address and the authentication information of the Microsoft HD Insight cluster to be used. For example, the address could be your_hdinsight_cluster_name.azurehdinsight.net and the authentication information is your Azure account name: ychen. The Studio uses this service to submit the Job to the HD Insight cluster.

In the Job result folder field, enter the location in which you want to store the execution result of a Job in the Azure Storage to be used. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HDInsight configuration</td>
<td>Enter the authentication information of the HD Insight cluster to be used.</td>
</tr>
</tbody>
</table>
| Windows Azure Storage configuration | Enter the address and the authentication information of the Azure Storage account to be used. In this configuration, you do not define where to read or write your business data but define where to deploy your Job only. Therefore always use the Azure Blob Storage system for this configuration.

In the Container field, enter the name of the container to be used.

In the Deployment Blob field, enter the location in which you want to store the current Job and its dependent libraries in this Azure Storage account. |
| Database | Fill this field with the name of the database. |

- When you use the other distributions:

<table>
<thead>
<tr>
<th>Connection mode</th>
<th>Select a connection mode from the list. The options vary depending on the distribution you are using.</th>
</tr>
</thead>
</table>
| Hive server | Select the Hive server through which you want the Job using this component to execute queries on Hive.

This Hive server list is available only when the Hadoop distribution to be used such as HortonWorks Data Platform V1.2.0 (Bimota) supports HiveServer2. It allows you to select HiveServer2 (Hive 2), the server that better support concurrent connections of multiple clients than HiveServer (Hive 1).

For further information about HiveServer2, see https://cwiki.apache.org/confluence/display/Hive/Setting+Up+HiveServer2. |
<p>| Host | Database server IP address. |
| Port | Listening port number of DB server. |</p>
<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Fill this field with the name of the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>This field is not available when you select Embedded from the Connection mode list.</td>
</tr>
</tbody>
</table>

| **Username and Password** | DB user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |

| **Use kerberos authentication** | If you are accessing a Hive Metastore running with Kerberos security, select this check box and then enter the relevant parameters in the fields that appear.  
- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.  
Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the Force MapR ticket authentication check box and the Use Kerberos authentication check box clear, and then MapR should be able to automatically find that ticket on the fly.  
The values of the following parameters can be found in the hive-site.xml file of the Hive system to be used.  
1. **Hive principal** uses the value of hive.metastore.kerberos.principal. This is the service principal of the Hive Metastore.  
2. **HiveServer2 local user principal** uses the value of hive.server2.authentication.kerberos.principal.  
3. **HiveServer2 local user keytab** uses the value of hive.server2.authentication.kerberos.keytab  
4. **Metastore URL** uses the value of javax.jdo.option.ConnectionURL. This is the JDBC connection string to the Hive Metastore.  
5. **Driver class** uses the value of javax.jdo.option.ConnectionDriverName. This is the name of the driver for the JDBC connection.  
6. **Username** uses the value of javax.jdo.option.ConnectionUserName. This, as well as the Password parameter, is the user credential for connecting to the Hive Metastore.  
7. **Password** uses the value of javax.jdo.option.ConnectionPassword.  
For the other parameters that are displayed, please consult the Hadoop configuration files they belong to. For example, the **Namenode principal** can be found in |
the `hdfs-site.xml` file or the `hdfs-default.xml` file of the distribution you are using.
This check box is available depending on the Hadoop distribution you are connecting to.

| **Use a keytab to authenticate** | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used. |

| **Use SSL encryption** | Select this check box to enable the SSL or TLS encrypted connection.

Then in the fields that are displayed, provide the authentication information:

- In the **Trust store path** field, enter the path, or browse to the TrustStore file to be used. By default, the supported TrustStore types are **JKS** and **PKCS 12**.

- To enter the password, click the `[...]` button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

This feature is available only to the HiveServer2 in the **Standalone** mode of the following distributions:

- Hortonworks Data Platform 2.0 +
- Cloudera CDH4 +
- Pivotal HD 2.0 +
- Amazon EMR 4.0.0 + |

| **Set Resource Manager** | Select this check box and in the displayed field, enter the location of the ResourceManager of your distribution. For example, `tal-qall4.tale nd.lan:8050`.

Then you can continue to set the following parameters depending on the configuration of the Hadoop cluster to be used (if you leave the check box of a parameter clear, then at runtime, the configuration about this parameter in the Hadoop cluster to be used will be ignored): |
1. Select the **Set resourcemanager scheduler address** check box and enter the Scheduler address in the field that appears.

2. Select the **Set jobhistory address** check box and enter the location of the JobHistory server of the Hadoop cluster to be used. This allows the metrics information of the current Job to be stored in that JobHistory server.

3. Select the **Set staging directory** check box and enter this directory defined in your Hadoop cluster for temporary files created by running programs. Typically, this directory can be found under the `yarn.app.mapreduce.am.staging-dir` property in the configuration files such as `yarn-site.xml` or `mapred-site.xml` of your distribution.

4. Allocate proper memory volumes to the **Map** and the **Reduce** computations and the **ApplicationMaster** of YARN by selecting the **Set memory** check box in the **Advanced settings** view.

5. Select the **Set Hadoop user** check box and enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed.

6. Select the **Use datanode hostname** check box to allow the Job to access datanodes via their hostnames. This actually sets the `dfs.client.use.datanode.hostname` property to `true`. When connecting to a S3N filesystem, you must select this check box.

For further information about these parameters, see the documentation or contact the administrator of the Hadoop cluster to be used.


---

**Set NameNode URI**

Select this check box and in the displayed field, enter the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, assuming that you have chosen a machine called `masternode` as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS.

The other properties:

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add
other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th>Hive version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
</table>
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  - **View schema**: choose this option to view the schema only. |

1606
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th><strong>Built-In</strong></th>
<th>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Table Name</strong></th>
<th>Name of the table to be created.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action on table</strong></td>
<td>Select the action to be carried out for creating a table.</td>
</tr>
</tbody>
</table>

| **Format** | Select the data format to which the table to be created is dedicated. The available data formats vary depending on the version of the Hadoop distribution you are using.  
Note that when the file format to be used is **PARQUET**, you might be prompted to find the specific Parquet jar file and install it into the Studio.  
- When the connection mode to Hive is **Embedded**, the Job is run in your local machine and calls this jar installed in the Studio.  
- When the connection mode to Hive is **Standalone**, the Job is run in the server hosting Hive and this jar file is sent to the HDFS system of the cluster you are connecting to. Therefore, ensure that you have properly defined the NameNode URI in the corresponding field of the Basic settings view.  
This jar file can be downloaded from Apache's site. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inputformat class</strong> and <strong>Outputformat class</strong></td>
<td>These fields appear only when you have selected INPUTFORMAT and OUTPUTFORMAT from the Format list.</td>
</tr>
<tr>
<td><strong>HiveCreateTable</strong></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td></td>
</tr>
<tr>
<td><strong>These fields allow you to enter the name of the jar files to be used for the data formats not available in the Format list.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Storage class</strong></td>
<td></td>
</tr>
<tr>
<td>Enter the name of the storage handler to be used for creating a non-native table (Hive table stored and managed in other systems than Hive, for example, Cassandra or MongoDB). This field is available only when you have selected STORAGE from the Format list. For further information about a storage handler, see <a href="https://cwiki.apache.org/confluence/display/Hive/StorageHandlers">https://cwiki.apache.org/confluence/display/Hive/StorageHandlers</a>.</td>
<td></td>
</tr>
<tr>
<td><strong>Set partitions</strong></td>
<td></td>
</tr>
<tr>
<td>Select this check box to add partition columns to the table to be created. Once selecting it, you need to define the schema of the partition columns you need to add.</td>
<td></td>
</tr>
<tr>
<td><strong>Set file location</strong></td>
<td></td>
</tr>
<tr>
<td>If you want to create a Hive table in a directory other than the default one, select this check box and enter the directory in HDFS you want to use to hold the table content. This is typical useful when you need to create an external Hive table by selecting the Create an external table check box in the Advanced settings tab.</td>
<td></td>
</tr>
<tr>
<td><strong>Use S3 endpoint</strong></td>
<td></td>
</tr>
<tr>
<td>The Use S3 endpoint check box is displayed when you have selected the Set file location check box to create an external Hive table. Once this Use S3 endpoint check box is selected, you need to enter the following parameters in the fields that appear: • S3 bucket: enter the name of the bucket in which you need to create the table. • Access key and Secret key: enter the authentication information required to connect to the Amazon S3 bucket to be used. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. Note that the format of the S3 file is S3N (S3 Native Filesystem). Since a Hive table created in S3 is actually an external table, this Use S3 endpoint check box must be used with the Create an external table case being selected.</td>
<td></td>
</tr>
<tr>
<td><strong>Advanced settings</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Like table</strong></td>
<td></td>
</tr>
<tr>
<td>Select this check box and enter the name of the Hive table you want to copy. This allows you to copy the definition of an existing table without copying its data.</td>
<td></td>
</tr>
<tr>
<td><strong>HiveCreateTable</strong></td>
<td>For further information about the Like parameter, see Apache's information about Hive's Data Definition Language.</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Create an external table</strong></td>
<td>Select this check box to make the table to be created an external Hive table. This kind of Hive table leaves the raw data where it is if the data is in HDFS. An external table is usually the better choice for accessing shared data existing in a file system. For further information about an external Hive table, see Apache's documentation about Hive.</td>
</tr>
<tr>
<td><strong>Table comment</strong></td>
<td>Enter the description you want to use for the table to be created.</td>
</tr>
<tr>
<td><strong>As select</strong></td>
<td>Select this check box and enter the As select statement for creating a Hive table that is based on a Select statement.</td>
</tr>
<tr>
<td><strong>Set clustered_by or skewed_by statement</strong></td>
<td>Enter the Clustered by statement to cluster the data of a table or a partition into buckets, or/and enter the Skewed by statement to allow Hive to extract the heavily skewed data and put it into separate files. This is typically used for obtaining better performance during queries.</td>
</tr>
<tr>
<td><strong>SerDe properties</strong></td>
<td>If you are using the SerDe row format, you can add any custom SerDe properties to override the default ones used by the Hadoop engine of the Studio.</td>
</tr>
<tr>
<td><strong>Table properties</strong></td>
<td>Add any custom Hive table properties you want to override the default ones used by the Hadoop engine of the Studio.</td>
</tr>
<tr>
<td><strong>Temporary path</strong></td>
<td>If you do not want to set the Jobtracker and the NameNode when you execute the query select * from your_table_name, you need to set this temporary path. For example, /C:/select_all in Windows.</td>
</tr>
</tbody>
</table>
| **Hadoop properties** | Talend Studio uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.  
  - Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in. 

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache's Hadoop documentation on http://hadoop.apache.org/docs and then select the version of the documentation you want. For
demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).
- Apache also provides a page to list the Hive-related properties: [https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties).

**Hive properties**

*Talend Studio* uses a default configuration for its engine to perform operations in a Hive database. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones. For further information for Hive dedicated properties, see [https://cwiki.apache.org/confluence/display/Hive/AdminManual+Configuration](https://cwiki.apache.org/confluence/display/Hive/AdminManual+Configuration).

- If you need to use Tez to run your Hive Job, add `hive.execution.engine` to the *Properties* column and `Tez` to the *Value* column, enclosing both of these strings in double quotation marks.
- Note that if you are using the centrally stored metadata from the *Repository*, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the *Property type* from *Repository* to *Built-in*.

**Mapred job map memory mb** and **Mapred job reduce memory mb**

You can tune the map and reduce computations by selecting the *Set memory* check box to set proper memory allocations for the computations to be performed by the Hadoop system.

In that situation, you need to enter the values you need in the **Mapred job map memory mb** and the **Mapred job reduce memory mb** fields, respectively. By default, the values are both 1000 which are normally appropriate for running the computations.

The memory parameters to be set are **Map (in Mb)**, **Reduce (in Mb)** and **ApplicationMaster (in Mb)**. These fields allow you to dynamically allocate memory to the map and the reduce computations and the ApplicationMaster of YARN.

**Path separator in server**

Leave the default value of the *Path separator in server* as it is, unless you have changed the separator used by your Hadoop distribution’s host machine for its PATH variable or in other words, that separator is not a colon (:). In that situation, you must change this value to the one you are using in that host.

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.
## Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>QUERY</strong>: the query statement being processed. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component works standalone.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the Studio used to connect to a Hive database is operated on Windows, you must manually create a folder called <strong>tmp</strong> in the root of the disk where this Studio is installed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Row format</th>
<th><strong>Set Delimited row format</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Set SerDe row format</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Die on error</strong></td>
</tr>
</tbody>
</table>

| Dynamic settings | Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |
|                 | The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable. |
|                 | For examples on using dynamic parameters, see [Scenario: Reading data from databases through context-based dynamic connections](#) on page 2497 and [Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters](#) on page 489. |
For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th><strong>Prerequisites</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>The Hadoop distribution must be properly installed, so as to guarantee the interaction with <em>Talend Studio</em>. The following list presents MapR related information for example.</td>
</tr>
<tr>
<td>• Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under <code>MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native</code>. For example, the library for Windows is <code>\lib\native\MapRClient.dll</code> in the MapR client jar file. For further information, see the following link from MapR: <a href="http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr">http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr</a>.</td>
</tr>
<tr>
<td>Without adding the specified library or libraries, you may encounter the following error: <code>no MapRClient in java.library.path</code>.</td>
</tr>
<tr>
<td>• Set the <code>-Djava.library.path</code> argument, for example, in the <em>Job Run VM arguments</em> area of the <em>Run/Debug</em> view in the [Preferences] dialog box in the <em>Window</em> menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.</td>
</tr>
<tr>
<td>For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.</td>
</tr>
</tbody>
</table>

### Related scenario

For a related scenario, see *Scenario: creating a partitioned Hive table* on page 1592.
tHiveInput

Extracts data from Hive and sends the data to the component that follows.

tHiveInput is the dedicated component to the Hive database (the Hive data warehouse system). It can execute a given HiveQL query in order to extract the data from Hive.

**tHiveInput Standard properties**

These properties are used to configure tHiveInput running in the Standard Job framework.

The Standard tHiveInput component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

Connection configuration:

- When you use this component with Google Dataproc:

<table>
<thead>
<tr>
<th><strong>Project identifier</strong></th>
<th>Enter the ID of your Google Cloud Platform project. If you are not certain about your project ID, check it in the Manage Resources page of your Google Cloud Platform services.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cluster identifier</strong></td>
<td>Enter the ID of your Dataproc cluster to be used.</td>
</tr>
<tr>
<td><strong>Region</strong></td>
<td>Enter the geographic zones in which the computing resources are used and your data is stored and processed. If you do not need to specify a particular region, leave the default value global. For further information about the available regions and the zones each region groups, see Regions and Zones.</td>
</tr>
<tr>
<td><strong>Google Storage staging bucket</strong></td>
<td>As a Talend Job expects its dependent jar files for execution, specify the Google Storage directory to which these jar files are transferred so that your Job can access these files at execution. The directory to be entered must end with a slash (/). If not existing, the directory is created on the fly but the bucket to be used must already exist.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Fill this field with the name of the database.</td>
</tr>
<tr>
<td><strong>Access Key</strong> and <strong>Secret Key</strong></td>
<td>Enter the authentication information obtained from Google for tHiveInput to read temporary data from Google Storage. These keys can be consulted on the Interoperable Access tab view under the Google Cloud Storage tab of the project from the Google APIs Console.</td>
</tr>
</tbody>
</table>
To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

For more information about the access key and secret key, go to [https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/](https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/) and see the description about developer keys.

<table>
<thead>
<tr>
<th>Provide Google Credentials in file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leave this check box clear, when you launch your Job from a given machine in which Google Cloud SDK has been installed and authorized to use your user account credentials to access Google Cloud Platform. In this situation, this machine is often your local machine. For further information about this Google Credentials file, see the administrator of your Google Cloud Platform or visit <a href="https://developers.google.com/storage/docs/reference/v1/getting-startedv1?hl=en/">Google Cloud Platform Auth Guide</a>.</td>
</tr>
</tbody>
</table>

**When you use this component with HDInsight:**

<table>
<thead>
<tr>
<th>WebHCat configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter the address and the authentication information of the Microsoft HD Insight cluster to be used. For example, the address could be your_hdinsight_cluster_name.azurehdinsight.net and the authentication information is your Azure account name: ychen. The Studio uses this service to submit the Job to the HD Insight cluster. In the <strong>Job result folder</strong> field, enter the location in which you want to store the execution result of a Job in the Azure Storage to be used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HDInsight configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter the authentication information of the HD Insight cluster to be used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Windows Azure Storage configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter the address and the authentication information of the Azure Storage account to be used. In this configuration, you do not define where to read or write your business data but define where to deploy your Job only. Therefore always use the Azure Blob Storage system for this configuration. In the <strong>Container</strong> field, enter the name of the container to be used. In the <strong>Deployment Blob</strong> field, enter the location in which you want to store the current Job and its dependent libraries in this Azure Storage account.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill this field with the name of the database.</td>
</tr>
</tbody>
</table>

**When you use the other distributions:**

<table>
<thead>
<tr>
<th>Connection mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select a connection mode from the list. The options vary depending on the distribution you are using.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hive server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the Hive server through which you want the Job using this component to execute queries on Hive.</td>
</tr>
</tbody>
</table>
This **Hive server** list is available only when the Hadoop distribution to be used such as **HortonWorks Data Platform V1.2.0 (Bimota)** supports HiveServer2. It allows you to select HiveServer2 (**Hive 2**), the server that better support concurrent connections of multiple clients than HiveServer (**Hive 1**).

For further information about HiveServer2, see [https://cwiki.apache.org/confluence/display/Hive/Setting+Up+HiveServer2](https://cwiki.apache.org/confluence/display/Hive/Setting+Up+HiveServer2).

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Fill this field with the name of the database.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Use kerberos authentication</strong></td>
<td>If you are accessing a Hive Metastore running with Kerberos security, select this check box and then enter the relevant parameters in the fields that appear.</td>
</tr>
<tr>
<td></td>
<td>• If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in <strong>Connecting to a security-enabled MapR</strong> on page 1652.</td>
</tr>
<tr>
<td></td>
<td>Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the <strong>Force MapR ticket authentication</strong> check box and the <strong>Use Kerberos authentication</strong> check box clear, and then MapR should be able to automatically find that ticket on the fly.</td>
</tr>
<tr>
<td></td>
<td>The values of the following parameters can be found in the <strong>hive-site.xml</strong> file of the Hive system to be used.</td>
</tr>
<tr>
<td>1. <strong>Hive principal</strong></td>
<td>uses the value of <strong>hive.metastore.kerberos.principal</strong>. This is the service principal of the Hive Metastore.</td>
</tr>
<tr>
<td>2. <strong>HiveServer2 local user principal</strong></td>
<td>uses the value of <strong>hive.server2.authentication.kerberos.principal</strong>.</td>
</tr>
<tr>
<td>3. <strong>HiveServer2 local user keytab</strong></td>
<td>uses the value of <strong>hive.server2.authentication.kerberos.keytab</strong>.</td>
</tr>
<tr>
<td>4. <strong>Metastore URL</strong></td>
<td>uses the value of <strong>javax.jdo.option.ConnectionURL</strong>. This is the JDBC connection string to the Hive Metastore.</td>
</tr>
</tbody>
</table>
5. **Driver class** uses the value of `javax.jdo.option.ConnectionDriverName`. This is the name of the driver for the JDBC connection.

6. **Username** uses the value of `javax.jdo.option.ConnectionUserName`. This, as well as the Password parameter, is the user credential for connecting to the Hive Metastore.

7. **Password** uses the value of `javax.jdo.option.ConnectionPassword`.

For the other parameters that are displayed, please consult the Hadoop configuration files they belong to. For example, the **Namenode principal** can be found in the `hdfs-site.xml` file or the `hdfs-default.xml` file of the distribution you are using.

This check box is available depending on the Hadoop distribution you are connecting to.

<table>
<thead>
<tr>
<th>Use a keytab to authenticate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the <strong>Use a keytab to authenticate</strong> check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the <strong>Principal</strong> field and the access path to the keytab file itself in the <strong>Keytab</strong> field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver. Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is <em>user1</em> and the principal to be used is <em>guest</em>; in this situation, ensure that <em>user1</em> has the right to read the keytab file to be used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use SSL encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to enable the SSL or TLS encrypted connection. Then in the fields that are displayed, provide the authentication information:</td>
</tr>
<tr>
<td>• In the <strong>Trust store path</strong> field, enter the path, or browse to the TrustStore file to be used. By default, the supported TrustStore types are <strong>JKS</strong> and <strong>PKCS 12</strong>.</td>
</tr>
<tr>
<td>• To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
</tbody>
</table>

This feature is available only to the HiveServer2 in the **Standalone** mode of the following distributions:

- Hortonworks Data Platform 2.0 +
- Cloudera CDH4 +
- Pivotal HD 2.0 +
- Amazon EMR 4.0.0 +
### Set Resource Manager

Select this check box and in the displayed field, enter the location of the ResourceManager of your distribution. For example, \texttt{tal-qa114.talend.lan:8050}.

Then you can continue to set the following parameters depending on the configuration of the Hadoop cluster to be used (if you leave the check box of a parameter clear, then at runtime, the configuration about this parameter in the Hadoop cluster to be used will be ignored):

1. Select the \texttt{Set resourcemanager scheduler address} check box and enter the Scheduler address in the field that appears.

2. Select the \texttt{Set jobhistory address} check box and enter the location of the JobHistory server of the Hadoop cluster to be used. This allows the metrics information of the current Job to be stored in that JobHistory server.

3. Select the \texttt{Set staging directory} check box and enter this directory defined in your Hadoop cluster for temporary files created by running programs. Typically, this directory can be found under the \texttt{yarn.app.mapreduce.am.staging-dir} property in the configuration files such as \texttt{yarn-site.xml} or \texttt{mapred-site.xml} of your distribution.

4. Allocate proper memory volumes to the Map and the Reduce computations and the ApplicationMaster of YARN by selecting the \texttt{Set memory} check box in the Advanced settings view.

5. Select the \texttt{Set Hadoop user} check box and enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed.

6. Select the \texttt{Use datanode hostname} check box to allow the Job to access datanodes via their hostnames. This actually sets the \texttt{dfs.client.use.datanode.hostname} property to \texttt{true}. When connecting to a S3N filesystem, you must select this check box.

For further information about these parameters, see the documentation or contact the administrator of the Hadoop cluster to be used.

For further information about the Hadoop Map/Reduce framework, see the Map/Reduce tutorial in Apache’s Hadoop documentation on \url{http://hadoop.apache.org}.

### Set NameNode URI

Select this check box and in the displayed field, enter the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, assuming that you have chosen a machine called \texttt{masternode}
as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS.


The other properties:

| Property type | Either Built-In or Repository.  
|               | **Built-In**: No property data stored centrally. 
|               | **Repository**: Select the repository file where the properties are stored. |

| Use an existing connection | Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined. |

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](https://help.talend.com).

| Distribution | Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration: |

- If available in this **Distribution** drop-down list, the **Microsoft HD Insight** option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center ([https://help.talend.com](https://help.talend.com)). |
• If you select **Amazon EMR**, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

• The **Custom** option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of Talend community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

| Hive version | Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using. |
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

| **Built-in** | The schema is created and stored locally for this component only. Related topic: see [Talend Studio User Guide](#). |
| **Repository** | The schema already exists and is stored in the Repository, hence can be reused. Related topic: see [Talend Studio User Guide](#). |
| **Table Name** | Name of the table to be processed. |
| **Query type** | Either **Built-in** or **Repository**. |
| **Built-in** | Fill in manually the query statement or build it graphically using SQLBuilder |
| **Repository** | Select the relevant query stored in the Repository. The Query field gets accordingly filled in. |
| **Guess Query** | Click the **Guess Query** button to generate the query which corresponds to your table schema in the Query field. |
| **Guess schema** | Click this button to retrieve the schema from the table. |
| **This query uses Parquet objects** | When available, select this check box to indicate that the table to be handled uses the Parquet format and thus make the component to call the required jar file. Note that when the file format to be used is **PARQUET**, you might be prompted to find the specific Parquet jar file and install it into the Studio.

- When the connection mode to Hive is **Embedded**, the Job is run in your local machine and calls this jar installed in the Studio.
- When the connection mode to Hive is **Standalone**, the Job is run in the server hosting Hive and this jar file is sent to the HDFS system of the cluster. |
you are connecting to. Therefore, ensure that you have properly defined the NameNode URI in the corresponding field of the **Basic settings** view.

This jar file can be downloaded from Apache's site. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

### Query

Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

For further information about the Hive query language, see https://cwiki.apache.org/confluence/display/Hive/LanguageManual.

#### Note:

Compressed data in the form of Gzip or Bzip2 can be processed through the query statements. For details, see https://cwiki.apache.org/confluence/display/Hive/CompressedStorage.

Hadoop provides different compression formats that help reduce the space needed for storing files and speed up data transfer. When reading a compressed file, the Studio needs to uncompress it before being able to feed it to the input flow.

### Execution engine

Select this check box and from the drop-down list, select the framework you need to use to run the Job.

This list is available only when you are using the **Embedded** mode for the Hive connection and the distribution you are working with is among the following ones:

- Hortonworks: V2.1 and V2.2.
- MapR: V4.0.1.
- Custom: this option allows you to connect to a distribution supporting Tez but not officially supported by *Talend*.

Before using Tez, ensure that the Hadoop cluster you are using supports Tez. You will need to configure the access to the relevant Tez libraries via the **Advanced settings** view of this component.

For further information about Hive on Tez, see Apache's related documentation in https://cwiki.apache.org/confluence/display/Hive/Hive+on+Tez. Some examples are presented there to show how Tez can be used to gain performance over MapReduce.

### Advanced settings

#### Tez lib

Select how the Tez libraries are accessed:

- **Auto install**: at runtime, the Job uploads and deploys the Tez libraries provided by the Studio into the
directory you specified in the **Install folder in HDFS** field, for example, `/tmp/usr/tez`.

If you have set the `tez.lib.uris` property in the properties table, this directory overrides the value of that property at runtime. But the other properties set in the properties table are still effective.

- **Use exist:** the Job accesses the Tez libraries already deployed in the Hadoop cluster to be used. You need to enter the path pointing to those libraries in the **Lib path (folder or file)** field.

- **Lib jar:** this table appears when you have selected **Auto install** from the Tez lib list and the distribution you are using is **Custom**. In this table, you need to add the Tez libraries to be uploaded.

### Temporary path

If you do not want to set the Jobtracker and the NameNode when you execute the query `select * from your_table_name`, you need to set this temporary path. For example, `/C:/select_all` in Windows.

### Trim all the String/Char columns

Select this check box to remove leading and trailing whitespace from all the String/Char columns.

### Trim column

Remove leading and trailing whitespace from defined columns.

**Note:**

Clear the **Trim all the String/Char columns** check box to enable **Trim column** in this field.

### Hadoop properties

**Talend Studio** uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- **Note** that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).
• Apache also provides a page to list the Hive-related properties: https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties.

**Hive properties**

_Talend Studio_ uses a default configuration for its engine to perform operations in a Hive database. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones. For further information for Hive dedicated properties, see https://cwiki.apache.org/confluence/display/Hive/AdminManual+Configuration.

• If you need to use Tez to run your Hive Job, add `hive.execution.engine` to the **Properties** column and `Tez` to the **Value** column, enclosing both of these strings in double quotation marks.

• Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.

**Mapred job map memory mb** and **Mapred job reduce memory mb**

You can tune the map and reduce computations by selecting the **Set memory** check box to set proper memory allocations for the computations to be performed by the Hadoop system.

In that situation, you need to enter the values you need in the **Mapred job map memory mb** and the **Mapred job reduce memory mb** fields, respectively. By default, the values are both 1000 which are normally appropriate for running the computations.

The memory parameters to be set are **Map (in Mb)**, **Reduce (in Mb)** and **ApplicationMaster (in Mb)**. These fields allow you to dynamically allocate memory to the map and the reduce computations and the ApplicationMaster of YARN.

**Path separator in server**

Leave the default value of the **Path separator in server** as it is, unless you have changed the separator used by your Hadoop distribution’s host machine for its PATH variable or in other words, that separator is not a colon (\`). In that situation, you must change this value to the one you are using in that host.

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. <strong>QUERY</strong>: the query statement being processed. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
</table>

**Usage**

**Usage rule**

This component offers the benefit of flexible DB queries and covers all possible Hive QL queries.

If the Studio used to connect to a Hive database is operated on Windows, you must manually create a folder called `tmp` in the root of the disk where this Studio is installed.

**HBase Configuration**

- **Note:**
  
  Available only when the **Use an existing connection** check box is clear

- **Store by HBase**

- **Zookeeper quorum**

- **Zookeeper client port**

- **Define the jars to register for HBase**

- **Register jar for HBase**

**Dynamic settings**

Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario:
### Reading data from different MySQL databases using dynamically loaded connection parameters

On page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL/hadoop/hadoop-VERSION/lib/native`. For example, the library for Windows is `lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

### Related scenarios

For a scenario about how an input component is used in a Job, see Scenario: Writing columns from a MySQL database to an output file using tMysqlInput on page 2492.

You need to keep in mind the parameters required by Hadoop, such as NameNode and Jobtracker, when configuring this component since the component needs to connect to a Hadoop distribution.
tHiveLoad

 Writes data of different formats into a given Hive table or to export data from a Hive table to a directory.

tHiveLoad connects to a given Hive database and copies or moves data into an existing Hive table or a directory you specify.

**tHiveLoad Standard properties**

These properties are used to configure tHiveLoad running in the Standard Job framework. The Standard tHiveLoad component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

Connection configuration:

- When you use this component with Google Dataproc:

<table>
<thead>
<tr>
<th><strong>Project identifier</strong></th>
<th>Enter the ID of your Google Cloud Platform project. If you are not certain about your project ID, check it in the Manage Resources page of your Google Cloud Platform services.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cluster identifier</strong></td>
<td>Enter the ID of your Dataproc cluster to be used.</td>
</tr>
<tr>
<td><strong>Region</strong></td>
<td>Enter the geographic zones in which the computing resources are used and your data is stored and processed. If you do not need to specify a particular region, leave the default value <em>global</em>. For further information about the available regions and the zones each region groups, see <a href="#">Regions and Zones</a>.</td>
</tr>
<tr>
<td><strong>Google Storage staging bucket</strong></td>
<td>As a Talend Job expects its dependent jar files for execution, specify the Google Storage directory to which these jar files are transferred so that your Job can access these files at execution. The directory to be entered must end with a slash (/). If not existing, the directory is created on the fly but the bucket to be used must already exist.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Fill this field with the name of the database.</td>
</tr>
<tr>
<td><strong>Provide Google Credentials in file</strong></td>
<td>Leave this check box clear, when you launch your Job from a given machine in which Google Cloud SDK has been installed and authorized to use your user account credentials to access Google Cloud Platform. In this situation, this machine is often your local machine.</td>
</tr>
</tbody>
</table>
### When you use this component with HDInsight:

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WebHCat configuration</strong></td>
<td>Enter the address and the authentication information of the Microsoft HD Insight cluster to be used. For example, the address could be your_hdinsight_cluster_name.azurehdinsight.net and the authentication information is your Azure account name: ychen. The Studio uses this service to submit the Job to the HD Insight cluster. In the <strong>Job result folder</strong> field, enter the location in which you want to store the execution result of a Job in the Azure Storage to be used.</td>
</tr>
<tr>
<td><strong>HDInsight configuration</strong></td>
<td>Enter the authentication information of the HD Insight cluster to be used.</td>
</tr>
<tr>
<td><strong>Windows Azure Storage configuration</strong></td>
<td>Enter the address and the authentication information of the Azure Storage account to be used. In this configuration, you do not define where to read or write your business data but define where to deploy your Job only. Therefore always use the Azure Blob Storage system for this configuration. In the <strong>Container</strong> field, enter the name of the container to be used. In the <strong>Deployment Blob</strong> field, enter the location in which you want to store the current Job and its dependent libraries in this Azure Storage account.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Fill this field with the name of the database.</td>
</tr>
</tbody>
</table>

### When you use the other distributions:

<table>
<thead>
<tr>
<th>Connection mode</th>
<th>Select a connection mode from the list. The options vary depending on the distribution you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hive server</strong></td>
<td>Select the Hive server through which you want the Job using this component to execute queries on Hive. This <strong>Hive server</strong> list is available only when the Hadoop distribution to be used such as <strong>HortonWorks Data Platform V1.2.0 (Bimota)</strong> supports HiveServer2. It allows you to select HiveServer2 (<strong>Hive 2</strong>), the server that better support concurrent connections of multiple clients than HiveServer (<strong>Hive 1</strong>). For further information about HiveServer2, see <a href="https://cwiki.apache.org/confluence/display/Hive/Setting+Up+HiveServer2">https://cwiki.apache.org/confluence/display/Hive/Setting+Up+HiveServer2</a>.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
</tbody>
</table>
**Database**

Fill this field with the name of the database.

**Note:**

This field is not available when you select **Embedded** from the **Connection mode** list.

---

**Username and Password**

DB user authentication data.

To enter the password, click the `[...]` button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

---

**Use kerberos authentication**

If you are accessing a Hive Metastore running with Kerberos security, select this check box and then enter the relevant parameters in the fields that appear.

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in **Connecting to a security-enabled MapR** on page 1652.

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

The values of the following parameters can be found in the *hive-site.xml* file of the Hive system to be used.

1. **Hive principal** uses the value of `hive.metastore.kerberos.principal`. This is the service principal of the Hive Metastore.

2. **HiveServer2 local user principal** uses the value of `hive.server2.authentication.kerberos.principal`.

3. **HiveServer2 local user keytab** uses the value of `hive.server2.authentication.kerberos.keytab`.

4. **Metastore URL** uses the value of `javax.jdo.option.ConnectionURL`. This is the JDBC connection string to the Hive Metastore.

5. **Driver class** uses the value of `javax.jdo.option.ConnectionDriverName`. This is the name of the driver for the JDBC connection.

6. **Username** uses the value of `javax.jdo.option.ConnectionUserName`. This, as well as the **Password** parameter, is the user credential for connecting to the Hive Metastore.

7. **Password** uses the value of `javax.jdo.option.ConnectionPassword`.

For the other parameters that are displayed, please consult the Hadoop configuration files they belong to. For example, the **Namenode principal** can be found in...
the `hdfs-site.xml` file or the `hdfs-default.xml` file of the distribution you are using.

This check box is available depending on the Hadoop distribution you are connecting to.

| **Use a keytab to authenticate** | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used. |

| **Use SSL encryption** | Select this check box to enable the SSL or TLS encrypted connection.

Then in the fields that are displayed, provide the authentication information:

- In the **Trust store path** field, enter the path, or browse to the TrustStore file to be used. By default, the supported TrustStore types are **JKS** and **PKCS 12**.
- To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

This feature is available only to the HiveServer2 in the **Standalone** mode of the following distributions:

- Hortonworks Data Platform 2.0 +
- Cloudera CDH4 +
- Pivotal HD 2.0 +
- Amazon EMR 4.0.0 +

| **Set Resource Manager** | Select this check box and in the displayed field, enter the location of the ResourceManager of your distribution. For example, `tal-qall4.talednd.lan:8050`.

Then you can continue to set the following parameters depending on the configuration of the Hadoop cluster to be used (if you leave the check box of a parameter clear, then at runtime, the configuration about this parameter in the Hadoop cluster to be used will be ignored): |
1. Select the **Set resourcemanager scheduler address** check box and enter the Scheduler address in the field that appears.

2. Select the **Set jobhistory address** check box and enter the location of the JobHistory server of the Hadoop cluster to be used. This allows the metrics information of the current Job to be stored in that JobHistory server.

3. Select the **Set staging directory** check box and enter this directory defined in your Hadoop cluster for temporary files created by running programs. Typically, this directory can be found under the `yarn.app.mapreduce.am.staging-dir` property in the configuration files such as `yarn-site.xml` or `mapred-site.xml` of your distribution.

4. Allocate proper memory volumes to the **Map** and the **Reduce** computations and the **ApplicationMaster** of YARN by selecting the **Set memory** check box in the **Advanced settings** view.

5. Select the **Set Hadoop user** check box and enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed.

6. Select the **Use datanode hostname** check box to allow the Job to access datanodes via their hostnames. This actually sets the `dfs.client.use.datanode.hostname` property to `true`. When connecting to a S3N filesystem, you must select this check box.

For further information about these parameters, see the documentation or contact the administrator of the Hadoop cluster to be used.


### Set NameNode URI

Select this check box and in the displayed field, enter the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, assuming that you have chosen a machine called `masternode` as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS.

The other properties:

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-In</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file where the properties are stored.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

   2. In the child level, use a dedicated connection component to read that registered database connection.

   For an example about how to share a database connection across Job levels, see **Talend Studio User Guide**.

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this **Distribution** drop-down list, the **Microsoft HD Insight** option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select **Amazon EMR**, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The **Custom** option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by **Talend**.

1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.
2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th>Hive version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
</table>
| Load action  | Select the action you need to carry for writing data into the specified destination.  
• When you select **LOAD**, you are moving or copying data from a directory you specify.  
• When you select **INSERT**, you are moving or copying data based on queries. |
| Execution engine | Select this check box and from the drop-down list, select the framework you need to use to perform the **INSERT** action.  
This list is available only when you are using the **Embedded** mode for the Hive connection and the |
distribution you are working with is among the following ones:

- Hortonworks: V2.1 and V2.2.
- MapR: V4.0.1.
- Custom: this option allows you connect to a distribution supporting Tez but not officially supported by Talend.

Before using Tez, ensure that the Hadoop cluster you are using supports Tez. You will need to configure the access to the relevant Tez libraries via the **Advanced settings** view of this component.

For further information about Hive on Tez, see Apache’s related documentation in https://cwiki.apache.org/confluence/display/Hive/Hive+on+Tez. Some examples are presented there to show how Tez can be used to gain performance over MapReduce.

<table>
<thead>
<tr>
<th>Target type</th>
<th>This drop-down list appears only when you have selected <strong>INSERT</strong> from the <strong>Load action</strong> list. Select from this list the type of the location you need to write data in.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- If you select <strong>Table</strong> as destination, you can still choose to append data to or overwrite the contents in the specified table.</td>
</tr>
<tr>
<td></td>
<td>- If you select <strong>Directory</strong> as destination, you are overwriting the contents in the specified directory.</td>
</tr>
</tbody>
</table>

| Table name  | Enter the name of the Hive table you need to write data in. Note that with the **INSERT** action, this field is available only when you have selected **Table** from the **Target type** list. |

<table>
<thead>
<tr>
<th>File path</th>
<th>Enter the directory you need to read data from or write data in, depending on the action you have selected from the <strong>Load action</strong> list.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- If you have selected <strong>LOAD</strong>: this is the path to the data you want to copy or move into the specified Hive table.</td>
</tr>
<tr>
<td></td>
<td>- If you have selected <strong>INSERT</strong>: this is the directory to which you want to export data from a Hive table. With this action, the <strong>File path</strong> field is available only when you have selected <strong>Directory</strong> from the <strong>Target type</strong> list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The target table uses the Parquet format</th>
<th>If the table in which you need to write data is a Parquet table, select this check box. Note that when the file format to be used is PARQUET, you might be prompted to find the specific Parquet jar file and install it into the Studio.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- When the connection mode to Hive is <strong>Embedded</strong>, the Job is run in your local machine and calls this jar installed in the Studio.</td>
</tr>
</tbody>
</table>
When the connection mode to Hive is **Standalone**, the Job is run in the server hosting Hive and this jar file is sent to the HDFS system of the cluster you are connecting to. Therefore, ensure that you have properly defined the NameNode URI in the corresponding field of the **Basic settings** view.

This jar file can be downloaded from Apache's site. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Then from the **Compression** list that appears, select the compression mode you need to use to handle the Parquet file. The default mode is **Uncompressed**.

<table>
<thead>
<tr>
<th>Action on file</th>
<th>Select the action to be carried out for writing data. This list is available only when the target is a Hive table; if the target is a directory, the action to be used is automatically <strong>OVERWRITE</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>This field appears when you have selected <strong>INSERT</strong> from the <strong>Load action</strong> list. Enter the appropriate query for selecting the data to be exported to the specified Hive table or directory.</td>
</tr>
<tr>
<td>Local</td>
<td>Select this check box to use the Hive <strong>LOCAL</strong> statement for accessing a local directory. Note that this local directory is actually in the machine in which the Job is run. Therefore, when the connection mode to Hive is <strong>Standalone</strong>, the Job is run in the machine where the Hive application is installed and thus this local directory is in that machine. This statement is used along with the directory you have defined in the <strong>File path</strong> field. Therefore, this <strong>Local</strong> check box is available only when the <strong>File path</strong> field is available.</td>
</tr>
<tr>
<td>Set partitions</td>
<td>Select this check box to use the Hive <strong>Partition</strong> clause in loading or inserting data in a Hive table. You need to enter the partition keys and their values to be used in the field that appears. For example, enter <code>country='US', state='CA'</code>. This makes a partition clause reading <strong>Partition (country='US', state='CA')</strong>, that is to say, a <strong>US and CA</strong> partition.</td>
</tr>
</tbody>
</table>
Also, it is recommended to select the **Create partition if not exist** check box that appears to ensure that you will not create a duplicate partition.

| Die on error | Select this check box to kill the Job when an error occurs. |

### Advanced settings

#### Tez lib

Select how the Tez libraries are accessed:

- **Auto install**: at runtime, the Job uploads and deploys the Tez libraries provided by the Studio into the directory you specified in the **Install folder in HDFS** field, for example, `/tmp/usr/tez`.

  If you have set the `tez.lib.uris` property in the properties table, this directory overrides the value of that property at runtime. But the other properties set in the properties table are still effective.

- **Use exist**: the Job accesses the Tez libraries already deployed in the Hadoop cluster to be used. You need to enter the path pointing to those libraries in the **Lib path (folder or file)** field.

- **Lib jar**: this table appears when you have selected **Auto install** from the **Tez lib** list and the distribution you are using is **Custom**. In this table, you need to add the Tez libraries to be uploaded.

#### Temporary path

If you do not want to set the Jobtracker and the NameNode when you execute the query `select * from your_table_name`, you need to set this temporary path. For example, `C:/select_all` in Windows.

#### Hadoop properties

**Talend Studio** uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such
Hive properties

Talend Studio uses a default configuration for its engine to perform operations in a Hive database. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones. For further information for Hive dedicated properties, see https://cwiki.apache.org/confluence/display/Hive/AdminManual+Configuration.

- If you need to use Tez to run your Hive Job, add hive.execution.engine to the Properties column and Tez to the Value column, enclosing both of these strings in double quotation marks.
- Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.

| Mapred job map memory mb and Mapred job reduce memory mb | You can tune the map and reduce computations by selecting the Set memory check box to set proper memory allocations for the computations to be performed by the Hadoop system. In that situation, you need to enter the values you need in the Mapred job map memory mb and the Mapred job reduce memory mb fields, respectively. By default, the values are both 1000 which are normally appropriate for running the computations. The memory parameters to be set are Map (in Mb), Reduce (in Mb) and ApplicationMaster (in Mb). These fields allow you to dynamically allocate memory to the map and the reduce computations and the ApplicationMaster of YARN. |
|-----------------------------------------------------------|
| Path separator in server | Leave the default value of the Path separator in server as it is, unless you have changed the separator used by your Hadoop distribution’s host machine for its PATH variable or in other words, that separator is not a colon (:). In that situation, you must change this value to the one you are using in that host. |
| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

Global Variables

| Global Variables | QUERY: the query statement being processed. This is a Flow variable and it returns a string. |
**Usage**

| Usage rule | This component works standalone and supports writing a wide range of data formats such as RC, ORC or AVRO.
|            | If the Studio used to connect to a Hive database is operated on Windows, you must manually create a folder called `tmp` in the root of the disk where this Studio is installed. |

| Dynamic settings | Click the `[]` button to add a row in the table and fill the `Code` field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

| Prerequisites | The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the `PATH` variable of that machine. According to MapR's documentation, the library or libraries of a MapR client corresponding to each... |
OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

**Related scenario**

For a related scenario, see [Scenario: creating a partitioned Hive table](#) on page 1592
### tHiveRow

Acts on the actual DB structure or on the data without handling data itself, depending on the nature of the query and the database.

tHiveRow executes the HiveQL query stated in the specified database. The row suffix means the component implements a flow in the Job design although it does not provide output.

The SQLBuilder tool helps you write your HiveQL statements easily.

This component can also perform queries in a HBase database once you activate its **Store by HBase** function.

### tHiveRow Standard properties

These properties are used to configure tHiveRow running in the Standard Job framework.

The Standard tHiveRow component belongs to the Big Data and the Databases families.

The component in this framework is generally available.

#### Basic settings

Connection configuration:

- When you use this component with Google Dataproc:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project identifier</strong></td>
<td>Enter the ID of your Google Cloud Platform project. If you are not certain about your project ID, check it in the Manage Resources page of your Google Cloud Platform services.</td>
</tr>
<tr>
<td><strong>Cluster identifier</strong></td>
<td>Enter the ID of your Dataproc cluster to be used.</td>
</tr>
<tr>
<td><strong>Region</strong></td>
<td>Enter the geographic zones in which the computing resources are used and your data is stored and processed. If you do not need to specify a particular region, leave the default value <code>global</code>. For further information about the available regions and the zones each region groups, see Regions and Zones.</td>
</tr>
<tr>
<td><strong>Google Storage staging bucket</strong></td>
<td>As a Talend Job expects its dependent jar files for execution, specify the Google Storage directory to which these jar files are transferred so that your Job can access these files at execution. The directory to be entered must end with a slash <code>/</code>. If not existing, the directory is created on the fly but the bucket to be used must already exist.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Fill this field with the name of the database.</td>
</tr>
<tr>
<td><strong>Provide Google Credentials in file</strong></td>
<td>Leave this check box clear, when you launch your Job from a given machine in which Google Cloud SDK has been installed and authorized to use your user account</td>
</tr>
</tbody>
</table>
credentials to access Google Cloud Platform. In this situation, this machine is often your local machine. For further information about this Google Credentials file, see the administrator of your Google Cloud Platform or visit Google Cloud Platform Auth Guide.

- When you use this component with HDInsight:

| **WebHCat configuration** | Enter the address and the authentication information of the Microsoft HD Insight cluster to be used. For example, the address could be your_hdsinsight_cluster_name.azurehdinsight.net and the authentication information is your Azure account name: ychen. The Studio uses this service to submit the Job to the HD Insight cluster.

In the **Job result folder** field, enter the location in which you want to store the execution result of a Job in the Azure Storage to be used. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HDInsight configuration</strong></td>
<td>Enter the authentication information of the HD Insight cluster to be used.</td>
</tr>
</tbody>
</table>
| **Windows Azure Storage configuration** | Enter the address and the authentication information of the Azure Storage account to be used. In this configuration, you do not define where to read or write your business data but define where to deploy your Job only. Therefore always use the Azure Blob Storage system for this configuration.

In the **Container** field, enter the name of the container to be used.

In the **Deployment Blob** field, enter the location in which you want to store the current Job and its dependent libraries in this Azure Storage account. |
| **Database** | Fill this field with the name of the database. |

- When you use the other distributions:

<table>
<thead>
<tr>
<th><strong>Connection mode</strong></th>
<th>Select a connection mode from the list. The options vary depending on the distribution you are using.</th>
</tr>
</thead>
</table>
| **Hive server** | Select the Hive server through which you want the Job using this component to execute queries on Hive.

This **Hive server** list is available only when the Hadoop distribution to be used such as *HortonWorks Data Platform V1.2.0 (Bimota)* supports HiveServer2. It allows you to select HiveServer2 (**Hive 2**), the server that better support concurrent connections of multiple clients than HiveServer (**Hive 1**).

For further information about HiveServer2, see https://cwiki.apache.org/confluence/display/Hive/Setting+Up+HiveServer2. |
<p>| <strong>Host</strong> | Database server IP address. |</p>
<table>
<thead>
<tr>
<th>Port</th>
<th>Listening port number of DB server.</th>
</tr>
</thead>
</table>
| **Database** | Fill this field with the name of the database.  

**Note:**  
This field is not available when you select **Embedded** from the **Connection mode** list. |
| **Username and Password** | DB user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| **Use kerberos authentication** | If you are accessing a Hive Metastore running with Kerberos security, select this check box and then enter the relevant parameters in the fields that appear.  
- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in **Connecting to a security-enabled MapR** on page 1652.  
Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.  
The values of the following parameters can be found in the **hive-site.xml** file of the Hive system to be used.  
1. **Hive principal** uses the value of `hive.metastore.kerberos.principal`. This is the service principal of the Hive Metastore.  
2. **HiveServer2 local user principal** uses the value of `hive.server2.authentication.kerberos.principal`.  
3. **HiveServer2 local user keytab** uses the value of `hive.server2.authentication.kerberos.keytab`.  
4. **Metastore URL** uses the value of `javax.jdo.option.ConnectionURL`. This is the JDBC connection string to the Hive Metastore.  
5. **Driver class** uses the value of `javax.jdo.option.ConnectionDriverName`. This is the name of the driver for the JDBC connection.  
6. **Username** uses the value of `javax.jdo.option.ConnectionUserName`. This, as well as the Password parameter, is the user credential for connecting to the Hive Metastore.  
7. **Password** uses the value of `javax.jdo.option.ConnectionPassword`.  
For the other parameters that are displayed, please consult the Hadoop configuration files they belong to. |
For example, the **Namenode principal** can be found in the *hdfs-site.xml* file or the *hdfs-default.xml* file of the distribution you are using.

This check box is available depending on the Hadoop distribution you are connecting to.

| **Use a keytab to authenticate** | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used. |

| **Use SSL encryption** | Select this check box to enable the SSL or TLS encrypted connection.

Then in the fields that are displayed, provide the authentication information:

- In the **Trust store path** field, enter the path, or browse to the TrustStore file to be used. By default, the supported TrustStore types are *JKS* and *PKCS 12*.

- To enter the password, click the `[...]` button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

This feature is available only to the HiveServer2 in the **Standalone** mode of the following distributions:

- Hortonworks Data Platform 2.0 +
- Cloudera CDH4 +
- Pivotal HD 2.0 +
- Amazon EMR 4.0.0 + |

| **Set Resource Manager** | Select this check box and in the displayed field, enter the location of the ResourceManager of your distribution. For example, `tal-qa114.taland.lan:8050`. Then you can continue to set the following parameters depending on the configuration of the Hadoop cluster to be used (if you leave the check box of a parameter clear, then at runtime, the configuration about this parameter in the Hadoop cluster to be used will be ignored): |

| | | |
1. Select the **Set resource manager scheduler address** check box and enter the Scheduler address in the field that appears.

2. Select the **Set job history address** check box and enter the location of the JobHistory server of the Hadoop cluster to be used. This allows the metrics information of the current Job to be stored in that JobHistory server.

3. Select the **Set staging directory** check box and enter this directory defined in your Hadoop cluster for temporary files created by running programs. Typically, this directory can be found under the `yarn.app.mapreduce.am.staging-dir` property in the configuration files such as `yarn-site.xml` or `mapred-site.xml` of your distribution.

4. Allocate proper memory volumes to the **Map** and the **Reduce** computations and the **Application Master** of YARN by selecting the **Set memory** check box in the **Advanced settings** view.

5. Select the **Set Hadoop user** check box and enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed.

6. Select the **Use datanode hostname** check box to allow the Job to access datanodes via their hostnames. This actually sets the `dfs.client.use.datanode.hostname` property to true. When connecting to a S3N filesystem, you must select this check box.

For further information about these parameters, see the documentation or contact the administrator of the Hadoop cluster to be used.


### Set NameNode URI

Select this check box and in the displayed field, enter the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, assuming that you have chosen a machine called `masternode` as the NameNode, then the location is `hdfs://masternode:portnumber`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS.

### Property type

Either **Built-In** or **Repository**.

**Built-In**: No property data stored centrally.

**Repository**: Select the repository file where the properties are stored.

### Use an existing connection

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see **Talend Studio User Guide**.

### Distribution

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this **Distribution** drop-down list, the **Microsoft HD Insight** option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center ([https://help.talend.com](https://help.talend.com)).

- If you select **Amazon EMR**, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

- The **Custom** option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by **Talend**.

1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.
2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th><strong>Hive version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
</table>
| **Execution engine** | Select this check box and from the drop-down list, select the framework you need to use to run the Job.  
This list is available only when you are using the **Embedded** mode for the Hive connection and the distribution you are working with is among the following ones:  
- Hortonworks: V2.1 and V2.2.  
- MapR: V4.0.1.  
- Custom: this option allows you connect to a distribution supporting Tez but not officially supported by **Talend**. |
Before using Tez, ensure that the Hadoop cluster you are using supports Tez. You will need to configure the access to the relevant Tez libraries via the Advanced settings view of this component.

For further information about Hive on Tez, see Apache’s related documentation in https://cwiki.apache.org/confluence/display/Hive/Hive+on+Tez. Some examples are presented there to show how Tez can be used to gain performance over MapReduce.

### Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Built-in</th>
<th>The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Table Name

Name of the table to be processed.

### Query type

Either **Built-in** or **Repository**.

**Built-in**: Fill in manually the query statement or build it graphically using SQLBuilder

**Repository**: Select the relevant query stored in the Repository. The Query field gets accordingly filled in.

### Guess Query

Click the **Guess Query** button to generate the query which corresponds to your table schema in the Query field.

### This query uses Parquet objects

When available, select this check box to indicate that the table to be handled uses the Parquet format and thus make the component to call the required jar file.
Note that when the file format to be used is **PARQUET**, you might be prompted to find the specific Parquet jar file and install it into the Studio.

- When the connection mode to Hive is **Embedded**, the Job is run in your local machine and calls this jar installed in the Studio.
- When the connection mode to Hive is **Standalone**, the Job is run in the server hosting Hive and this jar file is sent to the HDFS system of the cluster you are connecting to. Therefore, ensure that you have properly defined the NameNode URI in the corresponding field of the **Basic settings** view.

This jar file can be downloaded from Apache’s site. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

<table>
<thead>
<tr>
<th>Query</th>
<th>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition. For further information about the Hive query language, see <a href="https://cwiki.apache.org/confluence/display/Hive/LanguageManual">https://cwiki.apache.org/confluence/display/Hive/LanguageManual</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>Compressed data in the form of Gzip or Bzip2 can be processed through the query statements. For details, see <a href="https://cwiki.apache.org/confluence/display/Hive/CompressedStorage">https://cwiki.apache.org/confluence/display/Hive/CompressedStorage</a>. Hadoop provides different compression formats that help reduce the space needed for storing files and speed up data transfer. When reading a compressed file, the Studio needs to uncompress it before being able to feed it to the input flow.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>
| Store by HBase | Select this check box to display the parameters to be set to allow the Hive components to access HBase tables:  
- Once this access is configured, you will be able to use, in **tHiveRow** and **tHiveInput**, the Hive QL statements to read and write data in HBase.  
- If you are using the Kerberos authentication, you need to define the HBase related principals in the corresponding fields that are displayed.  
For further information about this access involving Hive and HBase, see Apache’s Hive documentation about Hive/HBase integration. |
| Zookeeper quorum | Type in the name or the URL of the Zookeeper service you use to coordinate the transaction between your... |
Studio and your database. Note that when you configure the Zookeeper, you might need to explicitly set the `zookeeper.znode.parent` property to define the path to the root znode that contains all the znodes created and used by your database; then select the Set Zookeeper znode parent check box to define this property.

**Zookeeper client port**
Type in the number of the client listening port of the Zookeeper service you are using.

**Define the jars to register for HBase**
Select this check box to display the Register jar for HBase table, in which you can register any missing jar file required by HBase, for example, the Hive Storage Handler, by default, registered along with your Hive installation.

**Register jar for HBase**
Click the [+] button to add rows to this table, then, in the Jar name column, select the jar file(s) to be registered and in the Jar path column, enter the path(s) pointing to that or those jar file(s).

### Advanced settings

**Tez lib**
Select how the Tez libraries are accessed:

- **Auto install**: at runtime, the Job uploads and deploys the Tez libraries provided by the Studio into the directory you specified in the Install folder in HDFS field, for example, `/tmp/usr/tez`.
  
  If you have set the `tez.lib.uris` property in the properties table, this directory overrides the value of that property at runtime. But the other properties set in the properties table are still effective.

- **Use exist**: the Job accesses the Tez libraries already deployed in the Hadoop cluster to be used. You need to enter the path pointing to those libraries in the Lib path (folder or file) field.

- **Lib jar**: this table appears when you have selected Auto install from the Tez lib list and the distribution you are using is Custom. In this table, you need to add the Tez libraries to be uploaded.

**Temporary path**
If you do not want to set the Jobtracker and the NameNode when you execute the query `select * from your_table_name`, you need to set this temporary path. For example, `/C:/select_all` in Windows.

**Propagate QUERY's recordset**
Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the use column list.

**Note:**
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY's recordset should be set to the
| **Hadoop properties** | *Talend Studio* uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache's Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).

- Apache also provides a page to list the Hive-related properties: [https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties).

| **Hive properties** | *Talend Studio* uses a default configuration for its engine to perform operations in a Hive database. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones. For further information for Hive dedicated properties, see [https://cwiki.apache.org/confluence/display/Hive/AdminManual+Configuration](https://cwiki.apache.org/confluence/display/Hive/AdminManual+Configuration).

- If you need to use Tez to run your Hive Job, add `hive.execution.engine` to the **Properties** column and **Tez** to the **Value** column, enclosing both of these strings in double quotation marks.

- Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.

| **Mapred job map memory mb** and **Mapred job reduce memory mb** | You can tune the map and reduce computations by selecting the **Set memory** check box to set proper memory allocations for the computations to be performed by the Hadoop system.

In that situation, you need to enter the values you need in the **Mapred job map memory mb** and the **Mapred job**.
**reduce memory mb** fields, respectively. By default, the values are both 1000 which are normally appropriate for running the computations.

**Path separator in server**

Leave the default value of the Path separator in server as it is, unless you have changed the separator used by your Hadoop distribution’s host machine for its PATH variable or in other words, that separator is not a colon (:). In that situation, you must change this value to the one you are using in that host.

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Query</th>
<th>ERROR_MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>QUERY</strong></td>
<td>the query statement being processed. This is a Flow variable and it returns a string.</td>
<td><strong>ERROR_MESSAGE</strong></td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

**Usage rule**

This component offers the benefit of flexible DB queries and covers all possible Hive QL queries.

tHiveRow can capture the Application_ID values and write them in the Job logs once you have activated Log4j and set the Log4j output level to Info for your Job involving tHiveRow.

- For further information about how to define the Log4j output level at an individual Job level, search for customizing log4j output level at runtime on Talend Help Center (https://help.talend.com).

- For further information about how to configure Log4j at the Studio level so as to apply the configuration to all Jobs, search for configuring Log4j on Talend Help Center (https://help.talend.com).

If the Studio used to connect to a Hive database is operated on Windows, you must manually create a folder called *tmp* in the root of the disk where this Studio is installed.
## Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

## Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with **Talend Studio**. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the **[Preferences]** dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the **Data viewer** to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.
Connecting to a security-enabled MapR

When designing a Job, set up the authentication configuration in the component you are using depending on how your MapR cluster is secured.

MapR supports the two following methods of authenticating a user and generating a MapR security ticket for this user: a username/password pair and Kerberos.

For further information about the MapR security mechanism, see MapR security architecture.
For a scenario about how to secure a MapR cluster, see Getting started with MapR security.

The different security scenarios you may face with your MapR cluster:

- When your MapR cluster is secured with Kerberos only, you only need to set up the typical Hadoop Kerberos configuration for your Job in the Studio.

- When your MapR cluster is secured with both the Kerberos mechanism and the MapR ticket security mechanism, you need to accordingly set up the configuration for both of them in your Job in the Studio.

  For details about how to configure the MapR ticket security mechanism in the Studio, see Setting up the MapR ticket authentication on page 1652.

- When your MapR cluster is secured with the MapR ticket security mechanism only, proceed as explained in Setting up the MapR ticket authentication on page 1652 to set up the MapR authentication configuration for your Job in the Studio.

You can also find an example of how to configure Kerberos authentication for a Talend Job in Talend Help Center (https://help.talend.com).

Although this example uses Cloudera for demonstration, the operations it describes are generic and thus applicable to MapR as well.

Setting up the MapR ticket authentication

Before you begin

- The MapR distribution you are using is from version 4.0.1 onwards and you have selected it as the cluster to connect to in the component to be configured.

- The MapR cluster has been properly installed and is running.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

  Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- This section explains only the authentication parameters to be used to connect to MapR. You still need to define the other parameters required by your Job.

  For further information, see the documentation about each component you are using.
**About this task**

In a Standard Job, you need to set up this configuration in the **Basic settings** tab of a Hadoop-related component to be used by your Job.

In the tab, you need to proceed as follows:

**Procedure**

1. Select the **Force MapR ticket authentication** check box to display the related parameters to be defined.

2. In the **Username** field, enter the username to be authenticated and in the **Password** field, specify the password used by this user.

   To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

   A MapR security ticket is generated for this user by MapR and stored in the machine where the Job you are configuring is executed.

3. If the **Group** field is available in this tab, you need to enter the name of the group to which the user to be authenticated belongs.

4. In the **Cluster name** field, enter the name of the MapR cluster you want to use this username to connect to.

   This cluster name can be found in the **mapr-clusters.conf** file located in /opt/mapr/conf of the cluster.

5. In the **Ticket duration** field, enter the length of time (in seconds) during which the ticket is valid.

**Using a custom MapR security configuration (optional)**

If the default security configuration of your MapR cluster has been changed, you need to configure the Job to be executed to take this custom security configuration into account.

MapR specifies its security configuration in the **mapr.login.conf** file located in /opt/mapr/conf of the cluster. For further information about this configuration file and the Java service it uses behind, see **mapr.login.conf** and JAAS.

**About this task**

To configure your Job, you need to define the related parameters in the **Basic settings** tab and the **Advanced settings** tab of the **Component** view of the component you want your Job to use to connect to MapR.

Proceed as follows to do the configuration:

**Procedure**

1. Verify what has been changed about this **mapr.login.conf** file.

   You should be able to obtain the related information from the administrator or the developer of your MapR cluster.

2. If the location of the MapR configuration files has been changed to somewhere else in the cluster, that is to say, the MapR Home directory has been changed, select the **Set the MapR Home directory** check box and enter the new Home directory. Otherwise, leave this check box clear and the default Home directory is used.
3. If the login module to be used in the `mapr.login.conf` file has been changed, select the `Specify the Hadoop login configuration` check box and enter the module to be called from the `mapr.login.conf` file. Otherwise, leave this check box clear and the default login module is used.

For example, enter `kerberos` to call the `hadoop_kerberos` module or `hybrid` to call the `hadoop_hybrid` module.

**Related scenarios**

For related topics, see:

- **Scenario: Combining two flows for selective output** on page 2558.
- **Procedure** on page 625.
- **Scenario: Removing and regenerating a MySQL table index** on page 2552.

You need to keep in mind the parameters required by Hadoop, such as NameNode and Jobtracker, when configuring this component since the component needs to connect to a Hadoop distribution.
tHSQLDbInput

Executes a DB query with a strictly defined order which must correspond to the schema definition and then it passes on the field list to the next component via a Main row link.

tHSQLDbInput reads a database and extracts fields based on a query.

**tHSQLDbInput Standard properties**

These properties are used to configure tHSQLDbInput running in the Standard Job framework.

The Standard tHSQLDbInput component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>![icon]</td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component <strong>Basic settings</strong> view. For more information about setting up and storing database connection parameters, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

**Running Mode**

Select on the list the Server Mode corresponding to your DB setup among the four propositions:

- HSQLDb Server, HSQLDb WebServer, HSQLDb In Process Persistent, HSQLDb In Memory.

**Use TLS/SSL sockets**

Select this check box to enable the secured mode if required.

**Host**

Database server IP address.

**Port**

Listening port number of DB server.

**Database Alias**

Alias name of the database.

**Username** and **Password**

DB user authentication data.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.
<table>
<thead>
<tr>
<th><strong>DB path</strong></th>
<th>Specify the directory to the database you want to connect to. This field is available only to the <strong>HSQLDb In Process Persistent</strong> running mode.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>By default, if the database you specify in this field does not exist, it will be created automatically. If you want to change this default setting, modify the connection parameter set in the <strong>Additional JDBC parameter</strong> field in the Advanced settings view</td>
</tr>
<tr>
<td><strong>Db name</strong></td>
<td>Enter the database name that you want to connect to. This field is available only to the <strong>HSQLDb In Process Persistent</strong> running mode and the <strong>HSQLDb In Memory</strong> running mode.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-in</strong>:</td>
<td>The schema is created and stored locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td><strong>Repository</strong>:</td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
<td></td>
</tr>
<tr>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td><strong>Query type and Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **Additional JDBC parameters** | Specify additional connection properties for the DB connection you are creating. When the running mode is **HSQLDb In Process Persistent**, you can set the connection property `ifexists=true` to allow connection |

1656
to an existing database only and avoid creating a new database.

**Trim all the String/Char columns**
Select this check box to remove leading and trailing whitespace from all the String/Char columns.

**Trim column**
Remove leading and trailing whitespace from defined columns.

**tStat Catcher Statistics**
Select this check box to collect log data at the component level.

### Global Variables

**Global Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>Indicates the number of lines processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>QUERY</td>
<td>Indicates the query to be processed. This is a Flow variable and it returns a string.</td>
</tr>
</tbody>
</table>

For further information about variables, see *Talend Studio User Guide*.

**Note:**
A Flow variable means it functions during the execution of a component while an After variable means it functions after the execution of a component.

### Usage

**Usage rule**
This component covers all possible SQL queries for HSQLDb databases.

**Connections**
Outgoing links (from this component to another):
- **Row**: Main; Iterate
- **Trigger**: Run if; On Component Ok; On Component Error; On Subjob Ok; On Subjob Error.

Incoming links (from one component to this one):
- **Row**: Iterate;
- **Trigger**: Run if; On Component Ok; On Component Error; On Subjob Ok; On Subjob Error.

For further information regarding connections, see *Talend Studio User Guide*.

**Limitation**
Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).
Related scenarios

For related topics, see:
tHSQLDbOutput

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tHSQLDbOutput writes, updates, makes changes or suppresses entries in a database.

**tHSQLDbOutput Standard properties**

These properties are used to configure tHSQLDbOutput running in the Standard Job framework.

The Standard tHSQLDbOutput component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><img src="image" alt="icon" /></td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component <strong>Basic settings</strong> view. For more information about setting up and storing database connection parameters, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td>Running Mode</td>
<td>Select on the list the Server Mode corresponding to your DB setup among the four propositions: <strong>HSQLDb Server</strong>, <strong>HSQLDb WebServer</strong>, <strong>HSQLDb In Process Persistent</strong>, <strong>HSQLDb In Memory</strong>.</td>
</tr>
<tr>
<td>Use TLS/SSL sockets</td>
<td>Select this check box to enable the secured mode if required.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td>DB path</td>
<td>Specify the directory to the database you want to connect to. This field is available only to the <strong>HSQLDb In Process Persistent</strong> running mode.</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>By default, if the database you specify in this field does not exist, it will be created automatically. If you want to change this default setting, modify the connection parameter set in the <strong>Additional JDBC parameter</strong> field in the <strong>Advanced settings</strong> view.</td>
</tr>
<tr>
<td>Db name</td>
<td>Enter the database name that you want to connect to. This field is available only to the <strong>HSQLDb In Process Persistent</strong> running mode and the <strong>HSQLDb In Memory</strong> running mode.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td><strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create a table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table if not exists</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop a table if exists and create</strong>: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear a table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td><strong>Action on data</strong></td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, Job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Make changes to existing entries.</td>
</tr>
<tr>
<td></td>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td><strong>Update or insert</strong>: Update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Remove entries corresponding to the input flow.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning</strong>:</td>
</tr>
</tbody>
</table>
|  | *It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the Update and Delete operations.*
To do that: Select the **Use field options** check box and then in the **Key in update column**, select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the **Key in delete column** for the Delete operation.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### Die on error

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

### Advanced settings

#### Additional JDBC parameters

Specify additional connection properties for the DB connection you are creating. When the running mode is **HSQLDb In Process Persistent**, you can set the connection property `ifexists=true` to allow connection to an existing database only and avoid creating a new database.
**Commit every**
Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.

**Additional Columns**
This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.

**Name:** Type in the name of the schema column to be altered or inserted as new column

**SQL expression:** Type in the SQL statement to be executed in order to alter or insert the relevant column data.

**Position:** Select Before, Replace or After following the action to be performed on the reference column.

**Reference column:** Type in a column of reference that the tDBOutput can use to place or replace the new or altered column.

**Use field options**
Select this check box to customize a request, especially when there is double action on data.

**Enable debug mode**
Select this check box to display each step during processing entries in a database.

**tStat Catcher Statistics**
Select this check box to collect log data at the component level.

### Global Variables

**Global Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_UPDATED</td>
<td>the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_INSERTED</td>
<td>the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_DELETED</td>
<td>the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_REJECTED</td>
<td>the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>QUERY</td>
<td>the query statement processed. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible. This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a MySQL database. It also allows you to create a reject flow using a <strong>Row &gt; Rejects</strong> link to filter data in error. For an example of <strong>tMySqlOutput</strong> in use, see <em>Scenario: Retrieving data in error with a Reject link</em> on page 2529.</th>
</tr>
</thead>
</table>

### Connections

Outgoing links (from this component to another):

- **Row**: Main; Reject
- **Trigger**: Run if; On Component Ok; On Component Error; On Subjob Ok; On Subjob Error.

Incoming links (from one component to this one):

- **Row**: Main;
- **Trigger**: Run if; On Component Ok; On Component Error; On Subjob Ok; On Subjob Error.

For further information regarding connections, see *Talend Studio User Guide*.

### Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in *Talend Help Center* (https://help.talend.com).

### Related scenarios

For related topics, see

- *Scenario: Inserting a column and altering data using tMysqlOutput* on page 2521.
tHSQLDbRow

Acts on the actual DB structure or on the data (although without handling data), depending on the nature of the query and the database. The SQLBuilder tool helps you write easily your SQL statements.

tHSQLDbRow is the specific component for this database query. It executes the SQL query stated onto the specified database. The row suffix means the component implements a flow in the job design although it doesn’t provide output.

**tHSQLDbRow Standard properties**

These properties are used to configure tHSQLDbRow running in the Standard Job framework.
The Standard tHSQLDbRow component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Running Mode**

Select on the list the Server Mode corresponding to your DB setup among the four propositions: HSQLDb Server, HSQLDb WebServer, HSQLDb In Process Persistent, HSQLDb In Memory.

**Use TLS/SSL sockets**

Select this check box to enable the secured mode if required.

**Host**

Database server IP address

**Port**

Listening port number of DB server.

**Database Alias**

Name of the database

**Username and Password**

DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**DB path**

Specify the directory to the database you want to connect to. This field is available only to the **HSQLDb In Process Persistent** running mode.

**Note:**
By default, if the database you specify in this field does not exist, it will be created automatically. If you want to change this default setting, modify the connection parameter set in the **Additional JDBC parameter** field in the **Advanced settings** view.

### Database

Enter the database name that you want to connect to. This field is available only to the **HSQLDb In Process Persistent** running mode and the **HSQLDb In Memory** running mode.

### Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

- **Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.
- **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

### Query type

Either **Built-in** or **Repository**.

- **Built-in**: Fill in manually the query statement or build it graphically using SQLBuilder.
- **Repository**: Select the relevant query stored in the Repository. The Query field gets accordingly filled in.

### Query

Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

### Die on error

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.
### Advanced settings

<table>
<thead>
<tr>
<th><strong>Advanced settings</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional JDBC parameters</strong></td>
<td>Specify additional connection properties for the DB connection you are creating. When the running mode is <strong>HSQLDb In Process Persistent</strong>, you can set the connection property <code>ifexists=true</code> to allow connection to an existing database only and avoid creating a new database.</td>
</tr>
<tr>
<td><strong>Propagate QUERY's recordset</strong></td>
<td>Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the <strong>use column</strong> list.</td>
</tr>
<tr>
<td><strong>Commit every</strong></td>
<td>Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Global Variables</strong></td>
<td><strong>QUERY</strong>: Indicates the query to be processed. This is a Flow variable and it returns a string. For further information about variables, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>

**Note:**

A Flow variable means it functions during the execution of a component while an After variable means it functions after the execution of a component.

### Usage

<table>
<thead>
<tr>
<th><strong>Usage</strong></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage rule</strong></td>
<td>This component offers the flexibility of the DB query and covers all possible SQL queries.</td>
</tr>
</tbody>
</table>
| **Connections** | Outgoing links (from this component to another): **Row**: Main; Reject; Iterate  
**Trigger**: Run if; On Component Ok; On Component Error; On Subjob Ok; On Subjob Error.  
Incoming links (from one component to this one):  
**Row**: Main; Iterate  
**Trigger**: Run if; On Component Ok; On Component Error; On Subjob Ok; On Subjob Error.  
For further information regarding connections, see **Talend Studio User Guide**. |
| **Limitation** | Due to license incompatibility, one or more JARs required to use this component are not provided. |
You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

**Related scenarios**

For related topics, see:

- **Procedure** on page 625.
- **Scenario: Removing and regenerating a MySQL table index** on page 2552.
**tHttpRequest**

Sends an HTTP request to the server and outputs the response information locally.

**tHttpRequest sends an HTTP request to the server end and gets the corresponding response information from the server end.**

**tHttpRequest Standard properties**

These properties are used to configure tHttpRequest running in the Standard Job framework.

The Standard tHttpRequest component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either *Built-in* or stored remotely in the *Repository*. Click *Edit schema* to make changes to the schema. If the current schema is of the *Repository* type, three options are available:  
  - *View schema*: choose this option to view the schema only.  
  - *Change to built-in property*: choose this option to change the schema to *Built-in* for local changes.  
  - *Update repository connection*: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select *No* upon completion and choose this schema metadata again in the [Repository Content] window. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the <em>Repository</em>. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td>Sync columns</td>
<td>Click this button to retrieve the schema from the preceding component.</td>
</tr>
<tr>
<td>URI</td>
<td>Type in the Uniform Resource Identifier (URI) that identifies the data resource on the server. A URI is similar to a URL, but more general.</td>
</tr>
<tr>
<td>Method</td>
<td>Select an HTTP method to define the action to be performed:</td>
</tr>
</tbody>
</table>
**HTTPRequest**

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Post</strong>: Sends data (for example HTML form data) to the server end.</td>
<td></td>
</tr>
<tr>
<td><strong>Get</strong>: Retrieves data from the server end.</td>
<td></td>
</tr>
</tbody>
</table>

**Post parameters from file**
- Browse to, or enter the path to the file that is used to provide parameters (request body) to the POST method.

**Write response content to file**
- Select this check box to save the HTTP response to a local file. You can either type in the file path in the input field or click the three-dot button to browse to the file path.

**Create directory if not exists**
- Select this check box to create the directory defined in the **Write response content to file** field if it does not exist.
- This check box appears only when the **Write response content to file** check box is selected and is cleared by default.

**Headers**
- Type in the name-value pair(s) for HTTP headers to define the parameters of the requested HTTP operation.
- **Key**: Fill in the name of the header field of an HTTP header.
- **Value**: Fill in the content of the header field of an HTTP header.
- For more information about definition of HTTP headers, please refer to: [en.wikipedia.org/wiki/List_of_HTTP_headers](http://en.wikipedia.org/wiki/List_of_HTTP_headers).

**Need authentication**
- Select this check box to fill in a user name and a password in the corresponding fields if authentication is needed:
- **user**: Fill in the user name for the authentication.
- **password**: Fill in the password for the authentication.
- To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**Die on error**
- Select the check box to stop the execution of the Job when an error occurs.
- Clear the check box to skip any rows on error and complete the process for error-free rows.

**Advanced settings**

**Set timeout**
- Select this check box to specify the connect and read timeout values in the following two fields:
  - **Connect timeout(s)**: Enter the connect timeout value in seconds. An exception will occur if the timeout expires before the connection can be established. The value of 0 indicates an infinite time out. By default, the connect timeout value is 30.
tHttpRequest

- **Read timeout(s):** Enter the read timeout value in seconds. An exception will occur if the timeout expires before there is data available for read. By default, the read timeout value is 0, which indicates an infinite time out.

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at a Job level and at each component level.</th>
</tr>
</thead>
</table>

**Global Variables**

**Global Variables**

- **ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

- **CONNECTED:** the result of whether a connection to the server established. This is an After variable and it returns a boolean.

- **RESPONSE_CODE:** the response code returned by the remote HTTP server. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.

**Usage**

**Usage rule**

This component can be used in sending HTTP requests to server and saving the response information. This component can be used as a standalone component.

**Scenario 1: Sending a HTTP request to the server and saving the response information to a local file**

This scenario describes a two-component Job that uses the **GET** method to retrieve information from the server end and writes the response to a local file as well as to the console.

**Linking the components**

**Procedure**

1. In the **Integration** perspective of the Studio, create a Job from the **Job Designs** node in the **Repository** tree view.

   For further information about how to create a Job, see the **Talend Studio User Guide**.

   For further information about how to create a Job, see the **Talend Studio User Guide**.
2. Drop the following components from the Palette onto the design workspace: **tHttpServletRequest** and **tLogRow**.

![Diagram of connections between tHttpServletRequest and tLogRow](image)

3. Connect the **tHttpServletRequest** component to the **tLogRow** component using a **Row > Main** connection.

### Configuring the GET request

**Procedure**

1. Double-click the **tHttpServletRequest** component to open its **Basic settings** view and define the component properties.

![Screenshot of tHttpServletRequest settings](image)

2. Fill in the **URI** field with "http://192.168.0.63:8081/testHttpRequest/build.xml". Note that this URI is for demonstration purposes only and it is not a live address.

3. From the **Method** list, select **GET**.

4. Select the **Write response content to file** check box and fill in the input field on the right with the file path by manual entry, **D:/test.txt** for this use case.

5. Select the **Need authentication** check box and fill in the user and password, both **tomcat** in this use case.

### Executing the Job

**About this task**

Then you can run this Job.

The **tLogRow** component is used to present the execution result of the Job.
Procedure

1. If you want to configure how the result is presented by tLogRow, double-click the component to open its Component view and in the Mode area, select the Table (print values in cells of a table) check box.

2. Press F6 to run this Job.

Results

Once done, the response information from the server is saved and displayed.

```
[statistics] connecting to socket on port 4011
[statistics] connected
logicRow_1
ResponseContent
hello world!
```

Scenario 2: Sending a POST request from a local JSON file

In this scenario, a four-component Job is used to read parameters from a given JSON file and send it in a POST request to a web site.

The JSON file to be used reads as follows:

```
{"echo":
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |
 |...
Linking the components

Procedure

1. In the Integration perspective of the Studio, create an empty Job, named httpRequestPostDemo for example, from the Job Designs node in the Repository tree view.
   For further information about how to create a Job, see the Talend Studio User Guide.
2. Drop tFileInputJSON, tFileOutputDelimited, tHttpRequest and tLogRow onto the workspace.
3. Connect tFileInputJSON to tHttpRequest using the Trigger > On Subjob Ok link.
4. Connect the other components using the Row > Main link.

Reading the JSON file

Procedure

1. Double-click tFileInputJSON to open its Component view.
   ![tFileInputJSON Component view](image)

2. Select JsonPath without loop from the Read By drop-down list.
3. Click the [...] button next to Edit schema to open the schema editor.
   ![Schema of tFileInputJSON](image)
4. Click the [+] button to add one row and name it, for example, to data.
5. Click OK to validate these changes and accept the propagation prompted by the pop-up dialog box.
6. In the Filename field, browse, or enter the path to the source JSON file in which the parameter to be sent is stored.
7. In the Mapping table, the data column you defined in the previous step in the component schema has been automatically added. In the JSONPath query column of this table, enter the JSON path, in double quotation marks, to extract the parameter to be sent. In this scenario, the path is echo[0].data.

**Writing the parameter to a flat file**

**Procedure**

1. Double-click tFileOutputDelimited to open its Component view.

   ![tFileOutputDelimited Component View](image)

2. In the File name field, browse, or enter the path to the flat file in which you want to write the extracted parameter. This file will be created if it does not exist. In this example, it is C:/tmp/postParamsFile.txt.

**Posting the parameter**

**Procedure**

1. Double-click tHttpRequest to open its Component view.

   ![tHttpRequest Component View](image)

2. In the URI field, enter the server address to which the parameter is to be sent. In this scenario, it is http://echo.itcuties.com/.

3. From the Method list, select POST.
4. In the **Post parameters from file** field, browse, or enter the path to the flat file that contains the parameter to be used. As defined earlier with the **tFileOutputDelimited** component, this path is `C:/tmp/postParamsFile.txt`.

**Executing the Job**

Press **F6** to run this Job.

The **tLogRow** component is used to present the execution result of the Job.

Once done, the **Run** view is opened automatically, where you can check the execution result.

```
[statistics] connecting to socket on port 3772
[statistics] connected
ITCuries h elloing ... "hello"
" ... "hello"
" ... "hello"
[statistics] disconnected
```

You can read that the site receiving the parameter returns answers.
tImpalaClose

Closes connection to an Impala database.

`tImpalaClose` closes an active connection to a given Impala database.

**tImpalaClose Standard properties**

These properties are used to configure `tImpalaClose` running in the Standard Job framework.

The Standard `tImpalaClose` component belongs to the Big Data family.

The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

| Component list | If there is more than one connection used in the Job, select `tImpalaConnection` from the list. |

**Advanced settings**

| `tStatCatcher Statistics` | Select this check box to collect the log data at a component level. |

**Global Variables**

| Global Variables | `ERROR_MESSAGE`: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the `Die on error` check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press `Ctrl + Space` to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

**Usage**

| Usage rule | This component is to be used along with the other Impala components, especially with `tImpalaConnection` as `tImpalaConnection` allows you to open a connection for the transaction which is underway. |

| Dynamic settings | Click the `[+]` button to add a row in the table and fill the `Code` field with a context variable to choose your database connection dynamically from multiple |
connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with **Talend Studio**. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under **MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native**. For example, the library for Windows is **\lib\native\MapRClient.dll** in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

  Without adding the specified library or libraries, you may encounter the following error: **no MapRClient in java.library.path**.

- Set the **-Djava.library.path** argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the **[Preferences]** dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

### Related scenarios

No scenario is available for the Standard version of this component yet.
tImpalaConnection

Establishes an Impala connection to be reused by other Impala components in your Job. tImpalaConnection opens a connection to an Impala database.

### tImpalaConnection Standard properties

These properties are used to configure tImpalaConnection running in the Standard Job framework. The Standard tImpalaConnection component belongs to the Big Data family. The component in this framework is available when you are using one of the Talend solutions with Big Data.

#### Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones require specific configuration:

- **If available in this Distribution drop-down list, the Microsoft HD Insight option** allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- **If you select Amazon EMR**, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- **The Custom option** allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different
Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the Import from existing version option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

Note:

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see Connecting to a custom Hadoop distribution on page 1589.

<table>
<thead>
<tr>
<th>Impala version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>DB server listening port.</td>
</tr>
<tr>
<td>Database</td>
<td>Fill this field with the name of the database.</td>
</tr>
<tr>
<td>Username</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td>Use kerberos authentication</td>
<td>If you are accessing an Impala system running with Kerberos security, select this check box and then enter the Kerberos principal of this Impala system.</td>
</tr>
<tr>
<td></td>
<td>• If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.</td>
</tr>
</tbody>
</table>
Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

This check box is available depending on the Hadoop distribution you are connecting to.

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect the log data at a component level.</th>
</tr>
</thead>
</table>

### Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |
|-------------------|-------------------------------------------------------------------|

### Usage

#### Usage rule

This component is generally used with other Impala components, particularly **tImpalaClose**.

#### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).
Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the `-Djava.library.path` argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

### Related scenario

This component is used in the similar way as a [tHiveConnection](#) component is. For further information, see Scenario: creating a partitioned Hive table on page 1592.
tImpalaCreateTable

Creates Impala tables that fit a wide range of Impala data formats.
tImpalaCreateTable connects to the Impala database to be used and creates an Impala table that is dedicated to data of the format you specify.

tImpalaCreateTable Standard properties

These properties are used to configure tImpalaCreateTable running in the Standard Job framework.
The Standard tImpalaCreateTable component belongs to the Big Data family.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

Use an existing connection

Select this check box and in the Component List, click the relevant connection component to reuse the connection details you already defined.

Note:
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

Distribution

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you
to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the Import from existing version option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

Note:

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see
### Connecting to a custom Hadoop distribution on page 1589.

<table>
<thead>
<tr>
<th><strong>Impala version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Fill this field with the name of the database.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td><strong>Use kerberos authentication</strong></td>
<td>If you are accessing an Impala system running with Kerberos security, select this check box and then enter the Kerberos principal of this Impala system.</td>
</tr>
<tr>
<td></td>
<td>- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.</td>
</tr>
<tr>
<td></td>
<td>Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the <strong>Force MapR ticket authentication</strong> check box and the <strong>Use Kerberos authentication</strong> check box clear, and then MapR should be able to automatically find that ticket on the fly.</td>
</tr>
<tr>
<td></td>
<td>This check box is available depending on the Hadoop distribution you are connecting to.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>- <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No upon completion and choose this schema metadata again in the [Repository Content] window.</strong></td>
</tr>
</tbody>
</table>
**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Name of the table to be created.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action on table</td>
<td>Select the action to be carried out for creating a table.</td>
</tr>
<tr>
<td>Format</td>
<td>Select the data format to which the table to be created is dedicated. The available data formats vary depending on the version of the Hadoop distribution you are using. Note that when the file format to be used is PARQUET, you might be prompted to find the specific Parquet jar file and install it into the Studio. • When the connection mode to Hive is Embedded, the Job is run in your local machine and calls this jar installed in the Studio. • When the connection mode to Hive is Standalone, the Job is run in the server hosting Hive and this jar file is sent to the HDFS system of the cluster you are connecting to. Therefore, ensure that you have properly defined the NameNode URI in the corresponding field of the Basic settings view. This jar file can be downloaded from Apache's site. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
<tr>
<td>Set partitions</td>
<td>Select this check box to add partition columns to the table to be created. Once selecting it, you need to define the schema of the partition columns you need to add.</td>
</tr>
<tr>
<td>Set file location</td>
<td>If you want to create an Impala table in a directory other than the default one, select this check box and enter the directory in HDFS you want to use to hold the table content. This is typical useful when you need to create an external Impala table by selecting the Create an external table check box in the Advanced settings tab.</td>
</tr>
<tr>
<td>Use S3 endpoint</td>
<td>The Use S3 endpoint check box is displayed when you have selected the Set file location check box to create an external Impala table.</td>
</tr>
</tbody>
</table>
Once this **Use S3 endpoint** check box is selected, you need to enter the following parameters in the fields that appear:

- **S3 bucket**: enter the name of the bucket in which you need to create the table.
- **Access key** and **Secret key**: enter the authentication information required to connect to the Amazon S3 bucket to be used.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

Note that the format of the S3 file is S3N (S3 Native Filesystem).

Since an Impala table created in S3 is actually an external table, this **Use S3 endpoint** check box must be used with the **Create an external table** case being selected.

### Advanced settings

| **Like table** | Select this check box and enter the name of the Impala table you want to copy. This allows you to copy the definition of an existing table without copying its data. For further information about the Like parameter, see Cloudera's information about Impala's Data Definition Language. |
| **Create an external table** | Select this check box to make the table to be created an external Impala table. This kind of Impala table leaves the raw data where it is if the data is in HDFS. An external table is usually the better choice for accessing shared data existing in a file system. For further information about an external Impala table, see Cloudera's documentation about Impala. |
| **Table comment** | Enter the description you want to use for the table to be created. |
| **As select** | Select this check box and enter the **As select** statement for creating an Impala table that is based on a **Select** statement. |
| **Table properties** | Add any custom Impala table properties you want to override the default ones used by the Hadoop engine of the Studio. |
| **tStatCatcher Statistics** | Select this check box to collect log data at the component level. |
### Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component works standalone.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row format</td>
<td>Set Delimited row format</td>
</tr>
<tr>
<td>Die on error</td>
<td></td>
</tr>
</tbody>
</table>

### Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine.
According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

### Related scenario

This component is used in the similar way as a [tHiveCreateTable](#) component is. For further information, see [Scenario: creating a partitioned Hive table](#) on page 1592.
tImpalaInput

Executes the select queries to extract the corresponding data and sends the data to the component that follows.

tImpalaInput is the dedicated component to the Impala database (the Impala data warehouse system). It executes the given Impala SQL query in order to extract the data of interest from Impala. It provides the SQLBuilder tool to help you write your Impala SQL statements easily.

**tImpalaInput Standard properties**

These properties are used to configure tImpalaInput running in the Standard Job framework. The Standard tImpalaInput component belongs to the Big Data family. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide.*

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:
• If available in this **Distribution** drop-down list, the **Microsoft HD Insight** option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

• If you select **Amazon EMR**, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

• The **Custom** option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.
For a step-by-step example about how to connect to a custom distribution and share this connection, see [Connecting to a custom Hadoop distribution](#) on page 1589.

<table>
<thead>
<tr>
<th><strong>Impala version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Fill this field with the name of the database.</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td><strong>Use kerberos authentication</strong></td>
<td>If you are accessing an Impala system running with Kerberos security, select this check box and then enter the Kerberos principal of this Impala system.</td>
</tr>
<tr>
<td></td>
<td>• If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in <a href="#">Connecting to a security-enabled MapR</a> on page 1652.</td>
</tr>
<tr>
<td></td>
<td>Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the <strong>Force MapR ticket authentication</strong> check box and the <strong>Use Kerberos authentication</strong> check box clear, and then MapR should be able to automatically find that ticket on the fly.</td>
</tr>
<tr>
<td></td>
<td>This check box is available depending on the Hadoop distribution you are connecting to.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the <strong>[Repository Content]</strong> window.</td>
</tr>
</tbody>
</table>
**Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

| **Table Name** | Name of the table to be processed. |
| **Query type** | Either **Built-in** or **Repository**. |

**Built-in**: Fill in manually the query statement or build it graphically using SQLBuilder

**Repository**: Select the relevant query stored in the Repository. The Query field gets accordingly filled in.

| **Guess Query** | Click the **Guess Query** button to generate the query which corresponds to your table schema in the Query field. |
| **Guess schema** | Click this button to retrieve the schema from the table. |

| **Query** | Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition. |

### Advanced settings

| **Trim all the String/Char columns** | Select this check box to remove leading and trailing whitespace from all the String/Char columns. |

| **Trim column** | Remove leading and trailing whitespace from defined columns. |

> **Note:**
> Clear the **Trim all the String/Char columns** check box to enable **Trim column** in this field.

| **tStatCatcher Statistics** | Select this check box to collect log data at the component level. |

### Global Variables

| **Global Variables** | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide.*

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the benefit of flexible DB queries and covers all possible Impala SQL queries.</th>
</tr>
</thead>
</table>

### Dynamic settings

Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

  Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the...
Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

Related scenarios

For a scenario about how an input component is used in a Job, see Scenario: Writing columns from a MySQL database to an output file using tMysqlInput on page 2492.
**tImpalaLoad**

Writes data of different formats into a given Impala table or to export data from an Impala table to a directory.

tImpalaLoad connects to a given Impala database and copies or moves data into an existing Impala table or a directory you specify.

**tImpalaLoad Standard properties**

These properties are used to configure tImpalaLoad running in the Standard Job framework.

The Standard tImpalaLoad component belongs to the Big Data family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see **Talend Studio User Guide**.

| Distribution | Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration: |
• If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

• If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

• The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the Import from existing version option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

Note:
In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.
For a step-by-step example about how to connect to a custom distribution and share this connection, see Connecting to a custom Hadoop distribution on page 1589.

<table>
<thead>
<tr>
<th><strong>Impala version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Fill this field with the name of the database.</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td><strong>Use kerberos authentication</strong></td>
<td>If you are accessing an Impala system running with Kerberos security, select this check box and then enter the Kerberos principal of this Impala system.</td>
</tr>
<tr>
<td></td>
<td>- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.</td>
</tr>
<tr>
<td></td>
<td>Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the Force MapR ticket authentication check box and the Use Kerberos authentication check box clear, and then MapR should be able to automatically find that ticket on the fly.</td>
</tr>
<tr>
<td></td>
<td>This check box is available depending on the Hadoop distribution you are connecting to.</td>
</tr>
<tr>
<td><strong>Load action</strong></td>
<td>Select the action you need to carry for writing data into the specified destination.</td>
</tr>
<tr>
<td></td>
<td>- When you select LOAD, you are moving or copying data from a directory you specify.</td>
</tr>
<tr>
<td></td>
<td>- When you select INSERT, you are moving or copying data based on queries.</td>
</tr>
<tr>
<td><strong>Target type</strong></td>
<td>This drop-down list appears only when you have selected INSERT from the Load action list. Select from this list the type of the location you need to write data in.</td>
</tr>
<tr>
<td></td>
<td>- If you select Table as destination, you can still choose to append data to or overwrite the contents in the specified table. This is the only option in the current release.</td>
</tr>
</tbody>
</table>
### Action
Select whether you want to **OVERWRITE** the old data already existing in the destination or only **APPEND** the new data to the existing one.

### Table name
Enter the name of the Hive table you need to write data in.

Note that with the **INSERT** action, this field is available only when you have selected **Table** from the **Target type** list.

### File path
Enter the directory you need to read data from.

### Query
This field appears when you have selected **INSERT** from the **Load action** list.

Enter the appropriate query for selecting the data to be exported to the specified Impala table or directory.

### Set partitions
Select this check box to use the Impala **Partition** clause in loading or inserting data in an Impala table. You need to enter the partition keys and their values to be used in the field that appears.

For example, enter `contry='US', state='CA'`. This makes a partition clause reading `Partition (contry='US', state='CA')`, that is to say, a **US and CA** partition.

Also, it is recommended to select the **Create partition if not exist** check box that appears to ensure that you will not create a duplicate partition.

### Die on error
Select this check box to kill the Job when an error occurs.

### Advanced settings

#### tStatCatcher Statistics
Select this check box to collect log data at the component level.

### Global Variables

**Global Variables**

<table>
<thead>
<tr>
<th><strong>QUERY</strong></th>
<th>the query statement being processed. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component works standalone.</th>
</tr>
</thead>
</table>

Dynamic settings

Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to
make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

Related scenario

This component is used in the similar way as a THiveLoad component is. For further information, see Scenario: creating a partitioned Hive table on page 1592.
tImpalaOutput

Executes the action defined on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tImpalaOutput connects to an Impala database (the Impala data warehouse system) and writes data in an Impala table.

**tImpalaOutput Standard properties**

These properties are used to configure tImpalaOutput running in the Standard Job framework.

The Standard tImpalaOutput component belongs to the Big Data family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

**Distribution**

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:
• If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

• If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

• The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the Import from existing version option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

Note:

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.
For a step-by-step example about how to connect to a custom distribution and share this connection, see Connecting to a custom Hadoop distribution on page 1589.

<table>
<thead>
<tr>
<th><strong>Impala version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Fill this field with the name of the database.</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>DB user authentication data.</td>
</tr>
</tbody>
</table>
| **Use kerberos authentication** | If you are accessing an Impala system running with Kerberos security, select this check box and then enter the Kerberos principal of this Impala system.  
  • If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.  
  Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the Force MapR ticket authentication check box and the Use Kerberos authentication check box clear, and then MapR should be able to automatically find that ticket on the fly.  
  This check box is available depending on the Hadoop distribution you are connecting to. |
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
  Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  • View schema: choose this option to view the schema only.  
  • Change to built-in property: choose this option to change the schema to Built-in for local changes.  
  • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the Repository Content window. |
| **Built-in:** | The schema is created and stored locally for this component only. Related topic: see **Talend Studio User Guide.** |
| **Repository:** | The schema already exists and is stored in the Repository, hence can be reused. Related topic: see **Talend Studio User Guide.** |

**Table Name**
Name of the table you need to write data in.

**Action**
Select whether you want to **OVERWRITE** the old data already existing in the destination or only **APPEND** the new data to the existing one.

**Extended insert**
Select this check box to combine multiple rows of data into one single INSERT action. This can speed up the insert operation.

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| **Global Variables** | **ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see **Talend Studio User Guide.** |

**Usage**

| **Usage rule** | This component offers the benefit of flexible DB queries and covers all possible Impala SQL queries. |
| **Dynamic settings** | Click the [*] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, |
when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with **Talend Studio**. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr). Without adding the specified library or libraries, you may encounter the following error: no `MapRClient` in `java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the [Preferences] dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the **Data viewer** to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

### Related scenarios

For a scenario about how an output component is used in a Job, see Scenario: Inserting a column and altering data using tMysqlOutput on page 2521.
ImpalaRow

Acts on the actual DB structure or on the data (although without handling data).

The SQLBuilder tool helps you write your Impala SQL statements easily. ImpalaRow is the dedicated component for this database. It executes the Impala SQL query stated in the specified database. The Row suffix means the component implements a flow in the Job design although it does not provide output.

ImpalaRow Standard properties

These properties are used to configure ImpalaRow running in the Standard Job framework.

The Standard ImpalaRow component belongs to the Big Data family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

Use an existing connection

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

Note:
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

Distribution

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:
• If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

• If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

• The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the Import from existing version option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

Note:
In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.
For a step-by-step example about how to connect to a custom distribution and share this connection, see [Connecting to a custom Hadoop distribution](#) on page 1589.

<table>
<thead>
<tr>
<th><strong>Impala version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Fill this field with the name of the database.</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>DB user authentication data.</td>
</tr>
</tbody>
</table>

**Use kerberos authentication** If you are accessing an Impala system running with Kerberos security, select this check box and then enter the Kerberos principal of this Impala system.

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in [Connecting to a security-enabled MapR](#) on page 1652.

  Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

  This check box is available depending on the Hadoop distribution you are connecting to.

**Schema and Edit Schema** A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.
### Built-in
The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

### Repository
The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Table Name</strong></th>
<th>Name of the table to be processed.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query type</strong></td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>Fill in manually the query statement or build it graphically using SQLBuilder</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</td>
</tr>
<tr>
<td><strong>Guess Query</strong></td>
<td>Click the Guess Query button to generate the query which corresponds to your table schema in the Query field.</td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row &gt; Rejects link.</td>
</tr>
</tbody>
</table>

### Advanced settings

**Propagate QUERY’s recordset**
Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the use column list.

**Note:**
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of Object and this component is usually followed by tParseRecordSet.

**tStatCatcher Statistics**
Select this check box to collect log data at the component level.

### Global Variables

**Global Variables**

- **QUERY**: the query statement being processed. This is a Flow variable and it returns a string.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

---

**Usage**

| Usage rule | This component offers the benefit of flexible DB queries and covers all possible Impala SQL queries. |

---

**Dynamic settings**

Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections on page 2497* and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489*. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

---

**Prerequisites**

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).
Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the **[Preferences]** dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the **Data viewer** to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

### Related scenarios

For related topics, see:

- **Scenario: Combining two flows for selective output** on page 2558.
- **Procedure** on page 625.
- **Scenario: Removing and regenerating a MySQL table index** on page 2552.
tInfiniteLoop

Execute a task or a Job automatically, based on a loop.
tInfiniteLoop runs an infinite loop on a task.

**tInfiniteLoop Standard properties**

These properties are used to configure tInfiniteLoop running in the Standard Job framework.
The Standard tInfiniteLoop component belongs to the Orchestration family.
The component in this framework is generally available.

**Basic settings**

| Wait at each iteration (in milliseconds) | Enter the time delay between iterations. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect the log data at a component level. |

**Global Variables**

| Global Variables | tStatCatcher Statistics: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. CURRENT_ITERATION: the sequence number of the current iteration. This is a Flow variable and it returns an integer. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

**Usage**

| Usage rule | tInfiniteLoop is an input component and requires an Iterate link to connect it to the following component. |

| Connections | Outgoing links (from this component to another): Row: Iterate |
**Related scenario**

For an example of the kind of scenario in which `tInfiniteLoop` might be used, see Procedure on page 1992, regarding the `tLoop` component.
tInformixBulkExec

Executes Insert operations in Informix databases.

tInformixOutputBulk and tInformixBulkExec are generally used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tInformixOutputBulkExec component. The advantage of using two components is that data can be transformed before it is loaded in the database.

**tInformixBulkExec Standard properties**

These properties are used to configure tInformixBulkExec running in the Standard Job framework.

The Standard tInformixBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in:</strong> No property data stored centrally.</td>
<td><strong>Repository:</strong> Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Execution Platform</td>
<td>Select the operating system you are using.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>DB server listening port.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Instance</strong></td>
<td>Name of the Informix instance to be used. This information can generally be found in the SQL hosts file.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
</tbody>
</table>
| **Action on table** | On the table defined, you can perform one of the following operations:  
  - **None**: No operation is carried out.  
  - **Drop and create a table**: The table is removed and created again.  
  - **Create a table**: The table does not exist and gets created.  
  - **Create a table if not exists**: The table is created if it does not exist.  
  - **Drop a table if exists and create**: The table is removed if it already exists and created again.  
  - **Clear a table**: The table content is deleted. |
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
  - **Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.  
  - **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*. When the schema to be reused has default values that are integers or functions, ensure that these default... |
values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

| Informix Directory | Informix installation directory, e.g. “C:\Program Files\IBM\IBM Informix Dynamic Server\11.50\”.
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data file</td>
<td>Name of the file to be loaded.</td>
</tr>
<tr>
<td>Action on data</td>
<td>On the data of the table defined, you can perform the following operations:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new data to the table. If duplicates are found, the job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Update the existing table data.</td>
</tr>
<tr>
<td></td>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td><strong>Update or insert</strong>: Update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Delete the entry data which corresponds to the input flow.</td>
</tr>
</tbody>
</table>

**Warning:**

*You must specify at least one key upon which the Update and Delete operations are to be based. It is possible to define the columns which should be used as the key from the schema, from both the Basic Settings and the Advanced Settings, to optimise these operations.*

### Advanced settings

**Additional JDBC parameters**

Specify additional connection properties for the DB connection you are creating. This option is not available
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field terminated by</td>
<td>Character, string or regular expression which separates the fields.</td>
</tr>
<tr>
<td>Set DBMONEY</td>
<td>Select this check box to define the decimal separator in the Decimal separator field.</td>
</tr>
<tr>
<td>Set DBDATE</td>
<td>Select the date format that you want to apply.</td>
</tr>
<tr>
<td>Rows Before Commit</td>
<td>Enter the number of rows to be processed before the commit.</td>
</tr>
<tr>
<td>Bad Rows Before Abort</td>
<td>Enter the number of rows in error at which point the Job should stop.</td>
</tr>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect the log data at component level.</td>
</tr>
<tr>
<td>Output</td>
<td>Where the output should go.</td>
</tr>
</tbody>
</table>

**Global Variables**

- **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.
- **NB_LINE_UPDATED**: the number of rows updated. This is an After variable and it returns an integer.
- **NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.
- **NB_LINE_DELETED**: the number of rows deleted. This is an After variable and it returns an integer.
- **QUERY**: the query statement processed. This is an After variable and it returns a string.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

- **Usage rule**: This component offers database query flexibility and covers all possible DB2 queries which may be required.
### Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Limitation

The database server/client must be installed on the same machine where the Studio is installed or where the Job using **tInformixBulkExec** is deployed, so that the component functions properly.

This component requires installation of its related jar files.

---

**Related scenario**

For a scenario in which **tInformixBulkExec** might be used, see:

- **Scenario: Inserting transformed data in MySQL database** on page 2537.
- **Scenario: Truncating and inserting file data into an Oracle database** on page 2752.
tInformixClose

Closes connection to Informix databases.

tInformixClose closes an active connection to a database.

**tInformixClose Standard properties**

These properties are used to configure tInformixClose running in the Standard Job framework.
The Standard tInformixClose component belongs to the Databases family.
The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>If there is more than one connection used in the Job, select tInformixConnection from the list.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics           | Select this check box to collect the log data at a component level. |

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is generally used as an input component. It requires an output component.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario:
Related scenario

This component is for use with tInformixConnection and tInformixRollback. They are generally used along with tInformixConnection as the latter allows you to open a connection for the transaction which is underway.

To see a scenario in which tInformixClose might be used, see tMysqlConnection on page 2475.
tInformixCommit

Makes a global commit just once instead of committing every row or batch of rows separately.
This component improves performance and is closely related to tInformixConnection and tInformixRollback. They are generally used to execute transactions together.
tInformixCommit validates data processed in a job from a connected database.

**tInformixCommit Standard properties**

These properties are used to configure tInformixCommit running in the Standard Job framework.
The Standard tInformixCommit component belongs to the Databases family.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>If there is more than one connection in the Job, select <strong>tInformixConnection</strong> from the list.</td>
</tr>
</tbody>
</table>
| Close connection | This check box is selected by default. It means that the database connection will be closed once the commit has been made. Clear the check box to continue using the connection once the component has completed its task. **Warning:**

If you are using a **Row > Main** type connection to link **tInformixCommit** to your Job, your data will be committed row by row. If this is the case, do not select this check box otherwise the connection will be closed before the commit of your first row is finalized. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect the log data at a component level. |
### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is generally used along with Informix components, particularly <code>tInformixConnection</code> and <code>tInformixRollback</code>.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see
- Scenario: Reading data from databases through context-based dynamic connections on page 2497
- Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

### Related Scenario

This component is for use with `tInformixConnection` and `tInformixRollback`. They are generally used along with `tInformixConnection` as the latter allows you to open a connection for the transaction which is underway.

To see a scenario in which `tInformixCommit` might be used, see Scenario: Inserting data in mother/daughter tables on page 2476.
**tInformixConnection**

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

`tInformixConnection` is closely related to `tInformixCommit` and `tInformixRollback`. They are generally used along with `tInformixConnection`, with `tInformixConnection` opening the connection for the transaction.

**tInformixConnection Standard properties**

These properties are used to configure `tInformixConnection` running in the Standard Job framework. The Standard `tInformixConnection` component belongs to the Databases family. The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>DB server listening port.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>Name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Instance</td>
<td>Name of the Informix instance to be used. This information can generally be found in the SQL hosts file.</td>
</tr>
</tbody>
</table>
### Additional JDBC parameters

Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the **Basic settings**.

### Use or register a shared DB Connection

Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the **Shared DB Connection Name** field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the **tRunJob** component. Using a shared connection together with a **tRunJob** component with either of these two options enabled will cause your Job to fail.

### Advanced settings

#### Use Transaction

Clear this check box when the database is configured in NO_LOG mode. If the check box is selected, you can choose whether to activate the Auto Commit option.

#### Auto Commit

Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

#### tStatCatcher Statistics

Select this check box to collect the log data at a component level.

### Usage

#### Usage rule

This component is generally used with other Informix components, particularly **tInformixCommit** and **tInformixRollback**.

#### Database Family

Databases/Informix
<table>
<thead>
<tr>
<th>Limitation</th>
<th>This component requires installation of its related jar files.</th>
</tr>
</thead>
</table>

**Related scenario**

For a scenario in which the `tnformixConnection`, might be used, see *Scenario: Inserting data in mother/daughter tables* on page 2476.
tInformixInput

Reads a database and extracts fields based on a query.

tInformixInput executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Main row link.

**tInformixInput Standard properties**

These properties are used to configure tInformixInput running in the Standard Job framework.

The Standard tInformixInput component belongs to the Databases family.

The component in this framework is generally available.

<i>Note:</i>

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored.</td>
</tr>
<tr>
<td></td>
<td>The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>DB server</td>
<td>Name of the database server</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button</td>
</tr>
<tr>
<td></td>
<td>next to the password field, and then in the pop-up dialog box enter</td>
</tr>
</tbody>
</table>
the password between double quotes and click **OK** to save the settings.

<table>
<thead>
<tr>
<th><strong>Schema and Edit schema</strong></th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>
| **Click Edit schema**      | Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. |
| **Query type and Query**   | Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition. |

### Global Variables

| **Global Variables** | **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. |
|------------------------|--------------------------------------------------------------------------------------------------|
For further information about variables, see *Talend Studio User Guide*.

## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible SQL queries for DB2 databases.</th>
</tr>
</thead>
</table>
| **Dynamic settings** | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489.

For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |

| Limitation | This component requires installation of its related jar files. |

## Related scenarios

For related topics, see:

See also scenario for **tContextLoad**: *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489.
tInformixOutput

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tInformixOutput writes, updates, makes changes or suppresses entries in a database.

**tInformixOutput Standard properties**

These properties are used to configure tInformixOutput running in the Standard Job framework.

The Standard tInformixOutput component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component.
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>DB server</td>
<td>Name of the database server</td>
</tr>
</tbody>
</table>
| Username and Password | DB user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Table         | Name of the table to be written. Note that only one table can be written at a time |
| Action on table | On the table defined, you can perform one of the following operations:  
None: No operation is carried out.  
Drop and create a table: The table is removed and created again.  
Create a table: The table does not exist and gets created.  
Create a table if not exists: The table is created if it does not exist.  
Drop a table if exists and create: The table is removed if it already exists and created again.  
Clear a table: The table content is deleted.  
Truncate table: Truncate the table.  

**Warning:**  
A commit operation will be carried out after the table is truncated. |
| Action on data | On the data of the table defined, you can perform:  
Insert: Add new entries to the table. If duplicates are found, Job stops.  
Update: Make changes to existing entries  
Insert or update: Insert a new record. If the record with the given reference already exists, an update would be made. |
**Update or insert**: Update the record with the given reference. If the record does not exist, a new record would be inserted.

**Delete**: Remove entries corresponding to the input flow.

**Warning**:

It is necessary to specify at least one column as a primary key on which the **Update** and **Delete** operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the **Update** and **Delete** operations. To do that: Select the **Use field options** check box and then in the **Key in update column**, select the check boxes next to the column names you want to use as a base for the **Update** operation. Do the same in the **Key in delete column** for the **Delete** operation.

---

**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

---

**Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.

---

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

---

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.

- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No**.
upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Die on error</th>
<th>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row &gt; Rejects link.</th>
</tr>
</thead>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>Additional JDBC parameters</th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the Use an existing connection check box in the Basic settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong> You can press Ctrl+Space to access a list of predefined global variables.</td>
</tr>
<tr>
<td>Commit every</td>
<td>Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at executions.</td>
</tr>
<tr>
<td>Additional Columns</td>
<td>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</td>
</tr>
<tr>
<td></td>
<td><strong>Name:</strong> Type in the name of the schema column to be altered or inserted as new column</td>
</tr>
<tr>
<td></td>
<td><strong>SQL expression:</strong> Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td></td>
<td><strong>Position:</strong> Select Before, Replace or After following the action to be performed on the reference column.</td>
</tr>
<tr>
<td></td>
<td><strong>Reference column:</strong> Type in a column of reference that the tDBOutput can use to place or replace the new or altered column.</td>
</tr>
<tr>
<td>Use field options</td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td>Enable debug mode</td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td>Use Batch</td>
<td>Select this check box to activate the batch mode for data processing.</td>
</tr>
<tr>
<td>Batch Size</td>
<td>Specify the number of records to be processed in each batch. This field appears only when the Use batch mode check box is selected.</td>
</tr>
</tbody>
</table>
## Optimize the batch insertion

Ensure the check box is selected, to optimize the insertion of batches of data.

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

## Global Variables

### NB_LINE

The number of rows processed. This is an After variable and it returns an integer.

### NB_LINE_UPDATED

The number of rows updated. This is an After variable and it returns an integer.

### NB_LINE_INSERTED

The number of rows inserted. This is an After variable and it returns an integer.

### NB_LINE_DELETED

The number of rows deleted. This is an After variable and it returns an integer.

### NB_LINE_REJECTED

The number of rows rejected. This is an After variable and it returns an integer.

### QUERY

The query statement processed. This is an After variable and it returns a string.

### ERROR_MESSAGE

The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

## Usage

### Usage rule

This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in an Informix database. It also allows you to create a reject flow using a Row > Rejects link to filter data in error. For an example of tMySqlOutput in use, see Scenario: Retrieving data in error with a Reject link on page 2529.

### Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases,
especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Limitation</th>
<th>This component requires installation of its related jar files.</th>
</tr>
</thead>
</table>

**Related scenarios**

For **tInformixOutput** related topics, see:

- Scenario: Inserting a column and altering data using **tMysqlOutput** on page 2521.
tInformixOutputBulk

Prepares the file to be used as a parameter in the INSERT query used to feed Informix databases.

tInformixOutputBulk and tInformixBulkExec are generally used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tInformixOutputBulkExec component. The advantage of using two components is that data can be transformed before it is loaded in the database.

Writes a file composed of columns, based on a defined delimiter and on Informix standards.

**tInformixOutputBulk Standard properties**

These properties are used to configure tInformixOutputBulk running in the Standard Job framework.

The Standard tInformixOutputBulk component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in:</strong> No property data stored centrally</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>File Name</td>
<td>Name of the file to be generated.</td>
</tr>
<tr>
<td>Append</td>
<td>Select this check box to append new rows to the end of the file.</td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-In:</strong> You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> You have already created the schema and stored it in the Repository. You can reuse it in various</td>
</tr>
</tbody>
</table>
InformixOutputBulk

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row separator</strong></td>
<td>String (ex: <code>\n</code> on Unix) to distinguish rows.</td>
</tr>
<tr>
<td><strong>Field separator</strong></td>
<td>Character, string or regular expression used to separate fields</td>
</tr>
<tr>
<td><strong>Set DBMONEY</strong></td>
<td>Select this box if you want to define the decimal separator in the corresponding field.</td>
</tr>
<tr>
<td><strong>Set DBDATE</strong></td>
<td>Select the date format that you want to apply.</td>
</tr>
<tr>
<td><strong>Create directory if not exists</strong></td>
<td>This check box is selected automatically. The option allows you to create a folder for the output file if it doesn’t already exist.</td>
</tr>
<tr>
<td><strong>Custom the flush buffer size</strong></td>
<td>Select this box in order to customize the memory size used to store the data temporarily. In the <strong>Row number</strong> field enter the number of rows at which point the memory should be freed.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
</tr>
</thead>
</table>
| **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is generally used along with tInformixBulkExec. Together, they improve performance levels when adding data to an Informix database.</th>
</tr>
</thead>
</table>
| **Dynamic settings** | Click the [*] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  
Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.  
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.  
For more information on Dynamic settings and context variables, see Talend Studio User Guide. |
| **Limitation** | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |
Related scenario

For a scenario in which `tlInformixOutputBulk` might be used, see:

- **Scenario: Inserting transformed data in MySQL database** on page 2537.
- **Scenario: Inserting data in bulk in MySQL database** on page 2544.
**tInformixOutputBulkExec**

Carries out Insert operations in Informix databases using the data provided.

*tInformixOutputBulk* and *tInformixBulkExec* are generally used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the *tInformixOutputBulkExec* component.

**tInformixOutputBulkExec Standard properties**

These properties are used to configure *tInformixOutputBulkExec* running in the Standard Job framework.

The Standard *tInformixOutputBulkExec* component belongs to the Databases family.

The component in this framework is generally available.

**i Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td>No properties stored centrally</td>
</tr>
<tr>
<td><strong>Repository:</strong> Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td><strong>Execution platform</strong></td>
<td>Select the operating system you are using.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**i Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>DB server listening port.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Instance</strong></td>
<td>Name of the Informix instance to be used. This information can generally be found in the <em>SQL hosts</em> file.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time and the table must already exist for the insert operation to be authorised.</td>
</tr>
</tbody>
</table>
| **Action on table** | On the table defined, you can perform one of the following operations:  
  **None**: No operation is carried out.  
  **Drop and create a table**: The table is removed and created again.  
  **Create a table**: The table does not exist and gets created.  
  **Create a table if not exists**: The table is created if it does not exist.  
  **Drop a table if exists and create**: The table is removed if it already exists and created again.  
  **Clear a table**: The table content is deleted. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields.  
  **Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*. |
| | **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*. |
When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Informix Directory
Informix installation directory, e.g. "C:\Program Files\IBM\IBM Informix Dynamic Server\11.50".

### Data file
Name of the file to be generated and loaded.

### Append
Select this check box to add rows to the end of the file.

### Action on data
Select the operation you want to perform:

- **Bulk insert**
- **Bulk update**

The details asked will be different according to the action chosen.

### Advanced settings

#### Additional JDBC parameters
Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the **Basic settings**.

**Note:**
You can press **Ctrl+Space** to access a list of predefined global variables.

- **Row separator**
  String (ex: ‘\n’ on Unix) to distinguish rows.

- **Fields terminated by**
  Character, string or regular expression used to separate the fields.

- **Set DBMONEY**
  Select this check box to define the decimal separator used in the corresponding field.

- **Set DBDATE**
  Select the date format you want to apply.
<table>
<thead>
<tr>
<th><strong>Rows Before Commit</strong></th>
<th>Enter the number of rows to be processed before the commit.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bad Rows Before Abort</strong></td>
<td>Enter the number of rows in error at which point the Job should stop.</td>
</tr>
<tr>
<td><strong>Create directory if not exists</strong></td>
<td>This check box is selected by default. It creates a directory to hold the output table if required.</td>
</tr>
<tr>
<td><strong>Custom the flush buffer size</strong></td>
<td>Select this box in order to customize the memory size used to store the data temporarily. In the Row number field enter the number of rows at which point the memory should be freed.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select the encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect the log data at a component level.</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>Where the output should go.</td>
</tr>
</tbody>
</table>

### Usage

**Usage rule**

This component is generally used when no particular transformation is required on the data to be inserted in the database.

**Dynamic settings**

Click the [*] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

**Limitation**

The database server/client must be installed on the same machine where the Studio is installed or where the Job using tInformixOutputBulkExec is deployed, so that the component functions properly.
Related scenario

For a scenario in which `tlInformixOutputBulkExec` might be used, see:

- Scenario: Inserting transformed data in MySQL database on page 2537.
- Scenario: Inserting data in bulk in MySQL database on page 2544.
**tInformixRollback**

Prevents involuntary transaction commits by canceling transactions in connected databases.

`tInformixRollback` is closely related to `tInformixCommit` and `tInformixConnection`. They are generally used together to execute transactions.

**tInformixRollback Standard properties**

These properties are used to configure `tInformixRollback` running in the Standard Job framework. The Standard `tInformixRollback` component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <code>tInformixConnection</code> component from the list if you plan to add more than one connection to the Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box if you want to continue to use the connection once the component has completed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect the log data at a component level. |

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component must be used with other Informix components, particularly <code>tInformixConnection</code> and <code>tInformixCommit</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Famille de composant</td>
<td>Databases/Informix</td>
</tr>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example,</td>
</tr>
</tbody>
</table>
when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related Scenario

For a scenario in which tInformixRollback might be used, see Scenario: Rollback from inserting data in mother/daughter tables on page 2480.
tInformixRow

Acts on the actual DB structure or on the data (although without handling data) thanks to the SQLBuilder that helps you write easily your SQL statements.

tInformixRow executes the SQL query stated onto the specified database. The Row suffix means the component implements a flow in the job design although it doesn’t provide output.

**tInformixRow Standard properties**

These properties are used to configure tInformixRow running in the Standard Job framework.
The Standard tInformixRow component belongs to the Databases family.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td><strong>1.</strong> In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
<td></td>
</tr>
<tr>
<td><strong>2.</strong> In the child level, use a dedicated connection component to read that registered database connection.</td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Query type</strong></td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
</tbody>
</table>
**Die on error**

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Additional JDBC parameters</strong></th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <strong>Use an existing connection</strong> check box in the <strong>Basic settings</strong>.</th>
</tr>
</thead>
</table>
| **Propagate QUERY’s recordset** | Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the **use column** list.  

**Note:**  
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of **Object** and this component is usually followed by **tParseRecordSet**. |
| **Use PreparedStatement** | Select this check box if you want to query the database using a PreparedStatement. In the **Set PreparedStatement Parameter** table, define the parameters represented by “?” in the SQL instruction of the **Query** field in the **Basic Settings** tab.  

- **Parameter Index**: Enter the parameter position in the SQL instruction.  
- **Parameter Type**: Enter the parameter type.  
- **Parameter Value**: Enter the parameter value.  

**Note:**  
This option is very useful if you need to execute the same query several times. Performance levels are increased |
| **Commit every** | Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions. |

| **tStat Catcher Statistics** | Select this check box to collect log data at the component level. |

### Global Variables

| **Global Variables** |  
- **QUERY**: the query statement being processed. This is a Flow variable and it returns a string.  
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions... |

1748
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
</table>

#### Dynamic settings

Click the [+ ] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Limitation</th>
<th>This component requires installation of its related jar files.</th>
</tr>
</thead>
</table>

### Related scenarios

For related topics, see:

- *Scenario: Combining two flows for selective output* on page 2558
- *Procedure* on page 625.
- *Scenario: Removing and regenerating a MySQL table index* on page 2552.
**tInformixSCD**

Tracks and shows changes which have been made to Informix SCD dedicated tables

tInformixSCD addresses Slowly Changing Dimension transformation needs, by regularly reading a data source and listing the modifications in an SCD dedicated table.

**tInformixSCD Standard properties**

These properties are used to configure tInformixSCD running in the Standard Job framework.
The Standard tInformixSCD component belongs to the Business Intelligence and the Databases families.
The component in this framework is generally available.

*Note:*
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <em>Apply</em>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <em>Built-in</em> or <em>Repository</em>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in:</strong> No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> Select the Repository file where properties are stored. The following fields are pre-filled in using fetched data</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <em>Component List</em> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

*Note:*
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the *Basic settings* view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.
### InformixSCD

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide.*

| **Host** | Database server IP address. |
| **Port** | DB server listening port. |
| **Database** | Name of the database. |
| **Schema** | Name of the schema. |
| **Username and Password** | User authentication information. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Instance** | Name of the Informix instance to be used. This information can generally be found in the SQL hosts file. |
| **Table** | Name of the table to be created |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window. |

- **Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide.*

- **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide.*

<p>| <strong>SCD Editor</strong> | The SCD editor helps to build and configure the data flow for slowly changing dimension outputs. For more information, see <em>SCD management methodology</em> on page 2566. |</p>
<table>
<thead>
<tr>
<th><strong>Use memory saving Mode</strong></th>
<th>Select this check box to improve system performance.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source keys include Null</strong></td>
<td>Select this check box to allow the source key columns to have Null values.</td>
</tr>
<tr>
<td><strong>Warning:</strong></td>
<td>Special attention should be paid to the uniqueness of the source key(s) values when this option is selected.</td>
</tr>
<tr>
<td><strong>Use Transaction</strong></td>
<td>Select this check box when the database is configured in NO_LOG mode.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>End date time details</strong></th>
<th>Specify the time value of the SCD end date time setting in the format of $HH:mm:ss$. The default value for this field is 12:00:00.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This field appears only when SCD <strong>Type 2</strong> is used and <strong>Fixed year value</strong> is selected for creating the SCD end date.</td>
</tr>
<tr>
<td><strong>Debug mode</strong></td>
<td>Select this check box to display each step of the process by which data is written in the database.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect the log data at a component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>NB_LINE_UPDATED</strong>: the number of rows updated. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_REJECTED</strong>: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>
### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is an output component. Consequently, it requires an input component and a connection of the Row &gt; Main type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Limitation</td>
<td>This component does not support using SCD type 0 together with other SCD types.</td>
</tr>
</tbody>
</table>

### Related scenario

For a scenario in which tInformixSCD might be used, see tMysqlSCD on page 2563.
tInformixSP

Centralises and calls multiple and complex queries in a database.
tInformixSP calls procedures stored in a database.

**tInformixSP Standard properties**

These properties are used to configure tInformixSP running in the Standard Job framework.
The Standard tInformixSP component belongs to the Databases family.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No properties stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see **Talend Studio User Guide**.
<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>User authentication information. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Instance</strong></td>
<td>Name of the Informix instance to be used. This information can generally be found in the SQL hosts file.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td><strong>SP Name</strong></td>
<td>Enter the exact name of the stored procedure (SP).</td>
</tr>
<tr>
<td><strong>Is Function / Return result in</strong></td>
<td>Select this check box if only one value must be returned. From the list, select the the schema column upon which the value to be obtained is based.</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td>Click the Plus button and select the various <strong>Schema Columns</strong> that will be required by the procedures. Note that the SP schema can hold more columns than there are parameters used in the procedure.</td>
</tr>
</tbody>
</table>
Select the **Type** of parameter:

**IN**: Input parameter.

**OUT**: Output parameter/return value.

**IN OUT**: Input parameters is to be returned as value, likely after modification through the procedure (function).

**RECORDSET**: Input parameters is to be returned as a set of values, rather than single value.

**Note:**

Check [Scenario: Inserting data in mother/daughter tables](#) on page 2476, if you want to analyze a set of records from a database table or DB query and return single records.

<table>
<thead>
<tr>
<th>Use Transaction</th>
<th>Clear this check box if the database is configured in the NO_LOG mode.</th>
</tr>
</thead>
</table>

### Advanced settings

**Additional JDBC parameters**

Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the **Basic settings**.

**tStatCatcher Statistics**

Select this check box to collect log data at a component level.

### Usage

**Usage rule**

This is an intermediary component. It can also be used as an entry component. In this case, only the entry parameters are authorized.

**Dynamic settings**

Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see [Scenario: Reading data from databases through context-based dynamic connections](#) on page 2497 and [Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters](#) on page 489.
For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

| Limitation | The stored procedure syntax must correspond to that of the database. This component requires installation of its related jar files. |

### Related scenarios

For related scenarios, see:

- **Scenario: Retrieving personal information using a stored procedure** on page 2452.
- **Scenario: Using tMysqlSP to find a State Label using a stored procedure** on page 2583.
- **Scenario: Checking number format using a stored procedure** on page 2809.
- **Executing a stored procedure using tMDMSP** on page 2190.

Also, see **Scenario: Inserting data in mother/daughter tables** on page 2476 if you want to analyse a set of records in a table or SQL query.
tIngresBulkExec

Inserts data in bulk to a table in the Ingres DBMS for performance gain.

tIngresOutputBulk and tIngresBulkExec are generally used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tIngresOutputBulkExec component. The advantage of using two components is that data can be transformed before it is loaded in the database.

**tIngresBulkExec Standard properties**

These properties are used to configure tIngresBulkExec running in the Standard Job framework.

The Standard tIngresBulkExec component belongs to the Databases family.

The component in this framework is generally available.

⚠️ **Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <em>Apply</em>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <em>Built-in</em> or <em>Repository</em>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be filled.</td>
</tr>
<tr>
<td><strong>VNode</strong></td>
<td>Name of the virtual node.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td>Actions that can be taken on the table defined:</td>
</tr>
<tr>
<td><strong>None</strong></td>
<td>No operation made to the table.</td>
</tr>
<tr>
<td><strong>Truncate</strong></td>
<td>Delete all the rows in the table and release the file space back to the operating system.</td>
</tr>
<tr>
<td><strong>File name</strong></td>
<td>Name of the file to be loaded.</td>
</tr>
<tr>
<td><strong>Warning:</strong></td>
<td>This file should be located on the same machine as the database server.</td>
</tr>
</tbody>
</table>
## Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

### Built-In

You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](https://help.talend.com).

### Repository

You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](https://help.talend.com).

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Delete Working Files After Use

Select this check box to delete the files that are created during the execution.

### Advanced settings

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Separator</td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td>Row Separator</td>
<td>String (ex: <code>\n</code> on Unix) to separate rows</td>
</tr>
<tr>
<td>Null Indicator</td>
<td>Value of the null indicator.</td>
</tr>
<tr>
<td>Session User</td>
<td>User of the defined session (the connection to the database).</td>
</tr>
<tr>
<td>Rollback</td>
<td>Enable or disable rollback.</td>
</tr>
<tr>
<td>On Error</td>
<td>Policy of error handling:</td>
</tr>
<tr>
<td></td>
<td><strong>Continue</strong>: Continue the execution.</td>
</tr>
<tr>
<td><strong>IngresBulkExec</strong></td>
<td><strong>Terminate</strong>: Terminate the execution.</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td><strong>Reject Row File</strong></td>
<td>Path and name of the file that holds the rejected rows. Available when <strong>Continue</strong> is selected from the <strong>On Error</strong> list.</td>
</tr>
<tr>
<td><strong>Error Count</strong></td>
<td>Number of errors to trigger the termination of the execution. Available when <strong>Terminate</strong> is selected from the <strong>On Error</strong> list.</td>
</tr>
<tr>
<td><strong>Allocation</strong></td>
<td>Number of pages initially allocated to the table or index.</td>
</tr>
<tr>
<td><strong>Extend</strong></td>
<td>Number of pages by which a table or index grows.</td>
</tr>
<tr>
<td><strong>Fill Factor</strong></td>
<td>Specify the percentage (from 1 to 100) of each primary data page that must be filled with rows, under ideal conditions. For example, if you specify a fillfactor of 40, the DBMS Server fills 40% of each of the primary data pages in the restructured table with rows.</td>
</tr>
<tr>
<td><strong>Min Pages/Max Pages</strong></td>
<td>Specify the minimum/maximum number of primary pages a hash table must have. The Min. pages and Max. pages must be at least 1.</td>
</tr>
<tr>
<td><strong>Leaf Fill</strong></td>
<td>A bulk copy from can specify a leaffill value. This clause specifies the percentage (from 1 to 100) of each B-tree leaf page that must be filled with rows during the copy. This clause can be used only on tables with a B-tree storage structure.</td>
</tr>
<tr>
<td><strong>Non Leaf Fill</strong></td>
<td>A bulk copy from can specify a nonleaffill value. This clause specifies the percentage (from 1 to 100) of each B-tree non-leaf index page that must be filled with rows during the copy. This clause can be used only on tables with a B-tree storage structure.</td>
</tr>
<tr>
<td><strong>Row Estimate</strong></td>
<td>Specify the estimated number of rows to be copied from a file to a table during a bulk copy operation.</td>
</tr>
<tr>
<td><strong>Trailing WhiteSpace</strong></td>
<td>Selected by default, this check box is designed to trim the trailing white spaces and applies only to such data types as VARCHAR, NVARCHAR and TEXT.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>List of the encoding schemes.</td>
</tr>
</tbody>
</table>
| **Output**          | Where to output the error message:  
<code>to console</code>: Message output to the console.  
<code>to global variable</code>: Message output to the global variable. |
| **tStatCatcher Statistics** | Select this check box to collect log data at the component level. |
Global Variables

| Global Variables | NB_LINE_DATA: the number of rows read. This is an After variable and it returns an integer. |
|                 | NB_LINE_BAD: the number of rows rejected. This is an After variable and it returns an integer. |
|                 | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |
|                 | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
|                 | To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. |
|                 | For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | Deployed along with tIngresOutputBulk, tIngresBulkExec feeds the given data in bulk to the Ingres database for performance gain. |
| Limitation | The database server/client must be installed on the same machine where the Studio is installed or where the Job using tIngresBulkExec is deployed, so that the component functions properly. |
|           | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |

Related scenarios

For related topics, see:

- Scenario: Loading data to a table in the Ingres DBMS on page 1782
**tIngresClose**

Closes the transaction committed in the connected Ingres database.

**tIngresClose Standard properties**

These properties are used to configure tIngresClose running in the Standard Job framework. The Standard tIngresClose component belongs to the Databases family. The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tIngresConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is to be used along with Ingres components, especially with tIngresConnection and tIngresCommit. |
| Dynamic settings | Click the [*] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-
Related scenarios

No scenario is available for the Standard version of this component yet.
tIngresCommit

Commits in one go, using a unique connection, a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

tIngresCommit validates the data processed through the Job into the connected database.

**tIngresCommit Standard properties**

These properties are used to configure tIngresCommit running in the Standard Job framework.

The Standard tIngresCommit component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tIngresConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Warning:**

*If you want to use a Row > Main connection to link tIngresCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.*

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tIngres* components, especially with the tIngresConnection and tIngresRollback components. |
### Dynamic settings

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>Field to fill with a context variable to choose database connection dynamically.</td>
</tr>
</tbody>
</table>

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenarios: Reading data from databases through context-based dynamic connections on page 2497 and Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Related scenario

For **tIngresCommit** related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476.
**tIngresConnection**

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

**tIngresConnection opens a connection to the database for a current transaction.**

**tIngresConnection Standard properties**

These properties are used to configure tIngresConnection running in the Standard Job framework.

The Standard tIngresConnection component belongs to the Databases and the ELT families.

The component in this framework is generally available.

<i>Note:</i>

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>&lt;ul&gt;&lt;li&gt;Built-in: No property data stored centrally.&lt;/li&gt;&lt;li&gt;Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.&lt;/li&gt;&lt;/ul&gt;</td>
</tr>
<tr>
<td>Server</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Use or register a shared DB Connection</td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the Shared DB Connection Name field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.</td>
</tr>
</tbody>
</table>
This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the **tRunJob** component. Using a shared connection together with a **tRunJob** component with either of these two options enabled will cause your Job to fail.

### Advanced settings

**Auto Commit**

Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

**tStatCatcher Statistics**

Select this check box to gather the job processing metadata at a Job level as well as at each component level.

### Usage

**Usage rule**

This component is more commonly used with other **tIngres** components, especially with the **tIngresCommit** and **tIngresRollback** components.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

### Related scenarios

For **tIngresConnection** related scenario, see Scenario: Loading data to a table in the Ingres DBMS on page 1782.
**tIngresInput**

Reads an Ingres database and extracts fields based on a query.

*tIngresInput* executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Main row link.

**tIngresInput Standard properties**

These properties are used to configure *tIngresInput* running in the Standard Job framework.

The Standard *tIngresInput* component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see *Dynamic database components* on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component <strong>Basic settings</strong> view. For more information about setting up and storing database connection parameters, see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component</td>
</tr>
</tbody>
</table>
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Server</th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
<td></td>
</tr>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td>Query type and Query</td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
</tbody>
</table>
### Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trim all the String/Char columns</td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</td>
</tr>
<tr>
<td>Trim column</td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>QUERY</td>
<td>the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it. For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

### Usage

| Usage rule                  | This component covers all possible SQL queries for Ingres databases. |

| Limitation                  | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)). |

### Related scenarios

For related topics, see:

See also the scenario for **tContextLoad**: Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
tIngresOutput

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tIngresOutput writes, updates, makes changes or suppresses entries in a database.

**tIngresOutput Standard properties**

These properties are used to configure tIngresOutput running in the Standard Job framework.
The Standard tIngresOutput component belongs to the Databases family.
The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view. For more information about setting up and storing database connection parameters, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component.
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td><strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create a table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table if not exists</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop a table if exists and create</strong>: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear a table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td>Action on data</td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, Job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Make changes to existing entries</td>
</tr>
<tr>
<td></td>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td><strong>Update or insert</strong>: Update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Remove entries corresponding to the input flow.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning</strong>: It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking</td>
</tr>
<tr>
<td><strong>Edit Schema</strong> and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the <strong>Advanced settings</strong> view where you can simultaneously define primary keys for the <strong>Update</strong> and <strong>Delete</strong> operations. To do that: Select the <strong>Use field options</strong> check box and then in the <strong>Key in update column</strong>, select the check boxes next to the column names you want to use as a base for the <strong>Update</strong> operation. Do the same in the <strong>Key in delete column</strong> for the <strong>Delete</strong> operation.</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
<tr>
<td><strong>Schema</strong> and <strong>Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
<td></td>
</tr>
</tbody>
</table>
| **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.  

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.  

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)). |
| **Die on error** | This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link. |
### Advanced settings

<table>
<thead>
<tr>
<th><strong>Commit every</strong></th>
<th>Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional Columns</strong></td>
<td>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Type in the name of the schema column to be altered or inserted as new column</td>
</tr>
<tr>
<td><strong>SQL expression</strong></td>
<td>Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td>Select Before, Replace or After following the action to be performed on the reference column.</td>
</tr>
<tr>
<td><strong>Reference column</strong></td>
<td>Type in a column of reference that the tDBOutput can use to place or replace the new or altered column.</td>
</tr>
<tr>
<td><strong>Use field options</strong></td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td><strong>Enable debug mode</strong></td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td><strong>Use Batch</strong></td>
<td>Select this check box to activate the batch mode for data processing.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>This check box is available only when you have selected the Insert, Update, or Delete option in the Action on data option.</td>
</tr>
<tr>
<td><strong>Batch Size</strong></td>
<td>Specify the number of records to be processed in each batch. This field appears only when the Use batch mode check box is selected.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_UPDATED</strong>: the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>
**Usage**

**Usage rule**

This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in an Ingres database. It also allows you to create a reject flow using a Row > Rejects link to filter data in error. For an example of tMySqlOutput in use, see **Scenario: Retrieving data in error with a Reject link** on page 2529.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

**Related scenarios**

For related topics, see:

- **Scenario: Inserting a column and altering data using tMySqlOutput** on page 2521.
tIngresOutputBulk

Prepares the file whose data is inserted in bulk to the Ingres DBMS for performance gain.

tIngresOutputBulk and tIngresBulkExec are generally used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tIngresOutputBulkExec component.

tIngresOutputBulk prepares a file with the schema defined and the data coming from the preceding component.

**tIngresOutputBulk Standard properties**

These properties are used to configure tIngresOutputBulk running in the Standard Job framework. The Standard tIngresOutputBulk component belongs to the Databases family.

The component in this framework is generally available.

- **Note:**
  
  This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

  For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>File Name</td>
<td>Name of the file to be generated.</td>
</tr>
<tr>
<td>Warning: This file is generated on the local machine or a shared folder on the LAN.</td>
<td></td>
</tr>
<tr>
<td>Append the File</td>
<td>Select this check box to add the new rows at the end of the file.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>
Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

### Advanced settings

<table>
<thead>
<tr>
<th>Field Separator</th>
<th>Character, string or regular expression to separate fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row Separator</td>
<td>String (ex: \n on Unix) to separate rows.</td>
</tr>
<tr>
<td>Include Header</td>
<td>Select this check box to include the column header in the file.</td>
</tr>
<tr>
<td>Encoding</td>
<td>List of encoding schemes.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Deployed along with <code>tIngresBulkExec</code>, <code>tIngresOutputBulk</code> is intended to save the incoming data to a file, whose data is then inserted in bulk to an Ingres database by <code>tIngresBulkExec</code> for performance gain.</th>
</tr>
</thead>
</table>

Limitation

<table>
<thead>
<tr>
<th>Limitation</th>
<th>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</th>
</tr>
</thead>
</table>

Related scenarios

For related topics, see:

- Scenario: Loading data to a table in the Ingres DBMS on page 1782,
**tIngresOutputBulkExec**

Inserts data in bulk to a table in the Ingres DBMS for performance gain.

tIngresOutputBulk and tIngresBulkExec are generally used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tIngresOutputBulkExec component.

tIngresOutputBulkExec prepared an output file and uses it to feed a table in the Ingres DBMS.

**tIngresOutputBulkExec Standard properties**

These properties are used to configure tIngresOutputBulkExec running in the Standard Job framework.

The Standard tIngresOutputBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in:</strong> No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> Select the repository file in which the properties are stored. The fields that</td>
</tr>
<tr>
<td></td>
<td>follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be filled.</td>
</tr>
<tr>
<td><strong>VNode</strong></td>
<td>Name of the virtual node.</td>
</tr>
<tr>
<td></td>
<td>The database server must be installed on the same machine where the Studio is installed or the</td>
</tr>
<tr>
<td></td>
<td>Job using tIngresOutputBulkExec is deployed.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td>Actions that can be taken on the table defined:</td>
</tr>
<tr>
<td></td>
<td><strong>None:</strong> No operation made to the table.</td>
</tr>
<tr>
<td></td>
<td><strong>Truncate:</strong> Delete all the rows in the table and release the file space back to the</td>
</tr>
<tr>
<td></td>
<td>operating system.</td>
</tr>
<tr>
<td><strong>File name</strong></td>
<td>Name of the file to be generated and loaded.</td>
</tr>
</tbody>
</table>

**Warning:**
This file is generated on the machine specified by the **VNode** field so it should be on the same machine as the database server.

### Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

### Delete Working Files After Use

Select this check box to delete the files that are created during the execution.

### Advanced settings

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Field Separator</strong></td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td><strong>Row Separator</strong></td>
<td>String (ex: <code>\n</code> on Unix) to separate rows</td>
</tr>
<tr>
<td><strong>On Error</strong></td>
<td>Policy of error handling:</td>
</tr>
<tr>
<td></td>
<td><strong>Continue</strong>: Continue the execution.</td>
</tr>
<tr>
<td></td>
<td><strong>Terminate</strong>: Terminate the execution.</td>
</tr>
<tr>
<td><strong>Reject Row File</strong></td>
<td>Path and name of the file that holds the rejected rows.</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Error Count</td>
<td>Number of errors to trigger the termination of the execution. Available when Terminate is selected from the On Error list.</td>
</tr>
<tr>
<td>Rollback</td>
<td>Enable or disable rollback.</td>
</tr>
<tr>
<td>Null Indicator</td>
<td>Value of the null indicator.</td>
</tr>
<tr>
<td>Session User</td>
<td>User of the defined session (the connection to the database).</td>
</tr>
<tr>
<td>Allocation</td>
<td>Number of pages initially allocated to the table or index.</td>
</tr>
<tr>
<td>Extend</td>
<td>Number of pages by which a table or index grows.</td>
</tr>
<tr>
<td>Fill Factor</td>
<td>Specify the percentage (from 1 to 100) of each primary data page that must be filled with rows, under ideal conditions. For example, if you specify a fill factor of 40, the DBMS Server fills 40% of each of the primary data pages in the restructured table with rows.</td>
</tr>
<tr>
<td>Min Pages/Max Pages</td>
<td>Specify the minimum/maximum number of primary pages a hash table must have. The Min. pages and Max. pages must be at least 1.</td>
</tr>
<tr>
<td>Leaf Fill</td>
<td>A bulk copy from can specify a leaf fill value. This clause specifies the percentage (from 1 to 100) of each B-tree leaf page that must be filled with rows during the copy. This clause can be used only on tables with a B-tree storage structure.</td>
</tr>
<tr>
<td>Non Leaf Fill</td>
<td>A bulk copy from can specify a nonleaf fill value. This clause specifies the percentage (from 1 to 100) of each B-tree non-leaf index page that must be filled with rows during the copy. This clause can be used only on tables with a B-tree storage structure.</td>
</tr>
<tr>
<td>Row Estimate</td>
<td>Specify the estimated number of rows to be copied from a file to a table during a bulk copy operation.</td>
</tr>
<tr>
<td>Trailing WhiteSpace</td>
<td>Selected by default, this check box is designed to trim the trailing white spaces and applies only to such data types as VARCHAR, NVARCHAR and TEXT.</td>
</tr>
<tr>
<td>Output</td>
<td>Where to output the error message: to console: Message output to the console. to global variable: Message output to the global variable.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
### Usage

| Usage rule | Usually deployed along with `tIngresConnection` or `tIngresRow`, `tIngresOutputBulkExec` prepares an output file and feeds its data in bulk to the Ingres DBMS for performance gain. |
| Limitation | The database server/client must be installed on the same machine where the Studio is installed or where the Job using `tIngresOutputBulkExec` is deployed, so that the component functions properly. Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |

### Scenario: Loading data to a table in the Ingres DBMS

In this scenario, a `tIngresOutputBulkExec` component is deployed to prepare an output file with the employee data from a .csv file and then use that output file to feed a table in an Ingres database.

### Dragging and dropping components

#### Procedure

1. Drop `tIngresConnection`, `tFileInputDelimited` and `tIngresOutputBulkExec` from the Palette onto the workspace.
2. Rename `tIngresOutputBulkExec` as `save_a_copy_and_load_to_DB`.
3. Link `tIngresConnection` to `tFileInputDelimited` using an OnSubjobOk trigger.
4. Link `tFileInputDelimited` to `tIngresOutputBulkExec` using a Row > Main connection.

### Configuring the components

#### Procedure

1. Double-click `tIngresConnection` to open its Basic settings view in the Component tab.
2. In the **Server** field, enter the address of the server where the Ingres DBMS resides, for example "localhost".

   Keep the default settings of the **Port** field.

3. In the **Database** field, enter the name of the Ingres database, for example "research".

4. In the **Username** and **Password** fields, enter the authentication credentials.

   A context variable is used for the password here. For more information on context variables, see *Talend Studio User Guide*.

5. Double-click **tFileInputDelimited** to open its **Basic settings** view in the **Component** tab.

6. Select the source file by clicking the [...] button next to the **File name/Stream** field.

7. Click the [...] button next to the **Edit schema** field to open the schema editor.
8. Click the [+] button to add four columns, for example name, age, job and dept, with the data type as string, Integer, string and string respectively.

Click OK to close the schema editor.

Click Yes on the pop-up window that asks whether to propagate the changes to the subsequent component.

Leave other default settings unchanged.

9. Double-click tIngresOutputBulkExec to open its Basic settings view in the Component tab.

10. In the Table field, enter the name of the table for data insertion.

11. In the VNode and Database fields, enter the names of the VNode and the database.

12. In the File Name field, enter the full path of the file that will hold the data of the source file.

**Executing the Job**

**Procedure**

1. Press Ctrl+S to save the Job.

2. Press F6 to run the Job.

As shown above, the employee data is written to the table employee in the database research on the node talendbj. Meanwhile, the output file employee_research.csv has been generated at C:/Users/talend/Desktop.
Related scenarios

For related topics, see:

- **Scenario: Inserting a column and altering data using tMysqlOutput** on page 2521.
**tIngresRollback**

Avoids to commit part of a transaction involuntarily by canceling the transaction committed in the connected database.

**tIngresRollback Standard properties**

These properties are used to configure tIngresRollback running in the Standard Job framework.

The Standard tIngresRollback component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tIngresConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tIngres* components, especially with the tIngresConnection and tIngresCommit components. |
| Dynamic settings | Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |
Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

For tIngresRollback related scenario, see Scenario: Rollback from inserting data in mother/daughter tables on page 2480.
tIngresRow

Acts on the actual DB structure or on the data (although without handling data) using the SQLBuilder tool to write easily your SQL statements.

tIngresRow executes the SQL query stated onto the specified database. The Row suffix means the component implements a flow in the job design although it doesn’t provide output.

**tIngresRow Standard properties**

These properties are used to configure tIngresRow running in the Standard Job framework.

The Standard tIngresRow component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
</tbody>
</table>
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Schema</strong> and <strong>Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
<td></td>
</tr>
<tr>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td><strong>Query type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: Fill in manually the query statement or build it graphically using SQLBuilder</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</td>
<td></td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
</tbody>
</table>
### Advanced Settings

<table>
<thead>
<tr>
<th><strong>Die on error</strong></th>
<th>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</th>
</tr>
</thead>
</table>

### Propagate QUERY’s recordset

Select this check box to insert the result of the query into a **COLUMN** of the current flow. Select this column from the **use column** list.

### Use PreparedStatement

Select this check box if you want to query the database using a PreparedStatement. In the **Set PreparedStatement Parameter** table, define the parameters represented by “?” in the SQL instruction of the **Query** field in the **Basic Settings** tab.

- **Parameter Index**: Enter the parameter position in the SQL instruction.
- **Parameter Type**: Enter the parameter type.
- **Parameter Value**: Enter the parameter value.

**Note:**

This option is very useful if you need to execute the same query several times. Performance levels are increased.

### Commit every

Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.

### tStat Catcher Statistics

Select this check box to collect log data at the component level.

### Global Variables

#### Global Variables

- **QUERY**: the query statement being processed. This is a Flow variable and it returns a string.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.
- A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
- To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
- For further information about variables, see *Talend Studio User Guide*. 

---

1790
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

Related scenarios

For related topics, see:
- Procedure on page 625.
- Scenario: Removing and regenerating a MySQL table index on page 2552.
tIngresSCD

Reflects and tracks changes in a dedicated Ingres SCD table.
tIngresSCD addresses Slowly Changing Dimension needs, reading regularly a source of data and logging the changes into a dedicated SCD table.

**tIngresSCD Standard properties**

These properties are used to configure tIngresSCD running in the Standard Job framework.
The Standard tIngresSCD component belongs to the Business Intelligence and the Databases families.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](#).

<table>
<thead>
<tr>
<th><strong>Property type</strong></th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in:</strong> No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the Repository file where properties are stored. The fields to follow are pre-filled in using fetched data.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>Server</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
</tbody>
</table>
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.  
  | **Built-in**: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.  
  | **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide. |
| **SCD Editor** | The SCD editor helps to build and configure the data flow for slowly changing dimension outputs. For more information, see SCD management methodology on page 2566. |
| **Use memory saving Mode** | Select this check box to maximize system performance. |
| **Source keys include Null** | Select this check box to allow the source key columns to have Null values. |
Warning:
Special attention should be paid to the uniqueness of the source key(s) values when this option is selected.

Die on error
This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

Advanced settings

End date time details
Specify the time value of the SCD end date time setting in the format of HH:mm:ss. The default value for this field is 12:00:00. This field appears only when SCD Type 2 is used and Fixed year value is selected for creating the SCD end date.

Debug mode
Select this check box to display each step during processing entries in a database.

tStatCatcher Statistics
Select this check box to collect log data at the component level.

Global Variables

Global Variables

NB_LINE_UPDATED: the number of rows updated. This is an After variable and it returns an integer.

NB_LINE_INSERTED: the number of rows inserted. This is an After variable and it returns an integer.

NB_LINE_REJECTED: the number of rows rejected. This is an After variable and it returns an integer.

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule
This component is used as Output component. It requires an Input component and Row main link as input.
### Limitation

| Limitation | This component does not support using SCD type 0 together with other SCD types. |

### Related scenario

For related scenarios, see [tMysqlSCD](#) on page 2563.
**tlInterbaseClose**

Closes the transaction committed in the connected Interbase database.

**tlInterbaseClose Standard properties**

These properties are used to configure tlInterbaseClose running in the Standard Job framework. The Standard tlInterbaseClose component belongs to the Databases family. The component in this framework is generally available.

**Note:**
- This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
- For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tlInterbaseConnection</strong> component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

**tStat Catcher Statistics**

Select this check box to collect log data at the component level.

**Usage**

**Usage rule**

This component is to be used along with Interbase components, especially with **tlInterbaseConnection** and **tlInterbaseCommit**.

**Dynamic settings**

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-**
Related scenarios

No scenario is available for the Standard version of this component yet.
**tInterbaseCommit**

Commits in one go a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

tInterbaseCommit validates the data processed through the Job into the connected DB.

**tInterbaseCommit Standard properties**

These properties are used to configure tInterbaseCommit running in the Standard Job framework.

The Standard tInterbaseCommit component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tInterbaseConnection</strong> component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task. <strong>Warning:</strong> If you want to use a Row &gt; Main connection to link <strong>tInterbaseCommit</strong> to your Job, your data will be committed row by row. In this case, do not select the <strong>Close connection</strong> check box or your connection will be closed before the end of your first row commit.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tInterbase components, especially with the **tInterbaseConnection** and **tInterbaseRollback** components. |
**Dynamic settings**

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Related scenario**

For **tInterbaseCommit** related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476.
tInterbaseConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

`tInterbaseConnection` opens a connection to the database for a current transaction.

**tInterbaseConnection Standard properties**

These properties are used to configure `tInterbaseConnection` running in the Standard Job framework.

The `tInterbaseConnection` component belongs to the Databases and the ELT families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Host name</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the <code>[...]</code> button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td>Use or register a shared DB Connection</td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the <strong>Shared DB Connection Name</strong> field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child. This option is incompatible with the <strong>Use dynamic job</strong> and <strong>Use an independent process to run subjob</strong> options.</td>
</tr>
</tbody>
</table>
of the `tRunJob` component. Using a shared connection together with a `tRunJob` component with either of these two options enabled will cause your Job to fail.

### Advanced settings

| **Auto Commit** | Select this check box to commit any changes to the database automatically upon the transaction. With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component. Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component. |
| **tStatCatcher Statistics** | Select this check box to gather the job processing metadata at a Job level as well as at each component level. |

### Usage

| **Usage rule** | This component is more commonly used with other `tInterbase*` components, especially with the `tInterbaseCommit` and `tInterbaseRollback` components. |
| **Limitation** | This component requires installation of its related jar files. |

### Related scenarios

For `tInterbaseConnection` related scenario, see `tMysqlConnection` on page 2475
tInterbaseInput

Reads an Interbase database and extracts fields based on a query.
tInterbaseInput executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Main row link.

tInterbaseInput Standard properties

These properties are used to configure tInterbaseInput running in the Standard Job framework.
The Standard tInterbaseInput component belongs to the Databases family.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

| Database | Select a type of database from the list and click Apply. |
| Property type | Either Built-in or Repository. |
| Built-in: No property data stored centrally. |
| Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved. |

Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.
For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

Use an existing connection

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

Note:
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component.
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: The schema is created and stored locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td><strong>Query type and Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
</tbody>
</table>
Advanced settings

<table>
<thead>
<tr>
<th><strong>Trim all the String/Char columns</strong></th>
<th>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trim column</strong></td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

Global Variables

| Global Variables | | |
|------------------|-------------------------------------------------------|
| **NB_LINE**      | the number of rows processed. This is an After variable and it returns an integer. |
| **QUERY**        | the query statement being processed. This is a Flow variable and it returns a string.          |
| **ERROR_MESSAGE**| the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. |

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible SQL queries for Interbase databases.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamic settings</strong></td>
<td>Click the <strong>[+]</strong> button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</td>
</tr>
</tbody>
</table>

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see [Scenario: Reading data from databases through context-](#).
Related scenarios

For related topics, see:

See also the related topic in **tContextLoad**: Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
tInterbaseOutput

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tInterbaseOutput writes, updates, makes changes or suppresses entries in a database.

**tInterbaseOutput Standard properties**

These properties are used to configure tInterbaseOutput running in the Standard Job framework.

The Standard tInterbaseOutput component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view. For more information about setting up and storing database connection parameters, see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. <strong>Note:</strong> When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to: 1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component.</td>
</tr>
</tbody>
</table>
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td><strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create a table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table if not exists</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop table if exists and create</strong>: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear a table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td>Action on data</td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, Job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Make changes to existing entries</td>
</tr>
<tr>
<td></td>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td><strong>Update or insert</strong>: Update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Remove entries corresponding to the input flow.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning</strong>: It is necessary to specify at least one column as a primary key on which the <strong>Update</strong> and <strong>Delete</strong> operations are based. You can do that by clicking <strong>Edit Schema</strong> and selecting the check box(es) next to the column(s) you want to set as primary key.</td>
</tr>
</tbody>
</table>
key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the **Update** and **Delete** operations. To do that: Select the **Use field options** check box and then in the **Key in update column**, select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the **Key in delete column** for the Delete operation.

<table>
<thead>
<tr>
<th>Clear data in table</th>
<th>Wipes out data from the selected table before action.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
<td>When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
<tr>
<td><strong>Click Edit schema to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</strong></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the <strong>[Repository Content]</strong> window.</td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the <strong>[Repository Content]</strong> window.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>
### Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Commit every</strong></td>
<td>Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.</td>
</tr>
<tr>
<td><strong>Additional Columns</strong></td>
<td>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</td>
</tr>
<tr>
<td></td>
<td><strong>Name:</strong> Type in the name of the schema column to be altered or inserted as new column</td>
</tr>
<tr>
<td></td>
<td><strong>SQL expression:</strong> Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td></td>
<td><strong>Position:</strong> Select <strong>Before</strong>, <strong>Replace</strong> or <strong>After</strong> following the action to be performed on the reference column.</td>
</tr>
<tr>
<td></td>
<td><strong>Reference column:</strong> Type in a column of reference that the tDBOutput can use to place or replace the new or altered column.</td>
</tr>
<tr>
<td><strong>Use field options</strong></td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td><strong>Enable debug mode</strong></td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td><strong>Use Batch</strong></td>
<td>Select this check box to activate the batch mode for data processing.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>This check box is available only when you have selected the <strong>Insert</strong>, <strong>Update</strong>, or <strong>Delete</strong> option in the Action on data option.</td>
</tr>
<tr>
<td><strong>Batch Size</strong></td>
<td>Specify the number of records to be processed in each batch. This field appears only when the Use batch mode check box is selected.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong></td>
<td>the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE UPDATED</strong></td>
<td>the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>
### Usage

#### Usage rule

This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible. This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in an Interbase database. It also allows you to create a reject flow using a **Row > Rejects** link to filter data in error. For an example of **tMySqlOutput** in use, see Scenario: Retrieving data in error with a Reject link on page 2529.

#### Dynamic settings

Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.
| Limitation | This component requires installation of its related jar files. |

**Related scenarios**

For related topics, see

- *Scenario: Inserting a column and altering data using tMysqlOutput* on page 2521.
**tlInterbaseRollback**

Avoids to commit part of a transaction involuntarily by canceling the transaction committed in the connected Interbase database.

**tlInterbaseRollback Standard properties**

These properties are used to configure tlInterbaseRollback running in the Standard Job framework.

The Standard tlInterbaseRollback component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tlInterbaseConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tInterbase* components, especially with the tlInterbaseConnection and tlInterbaseCommit components. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |
Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

For tInterbaseRollback related scenario, see Scenario: Rollback from inserting data in mother/daughter tables on page 2480.
tInterbaseRow

Acts on the actual database structure or on the data (although without handling data) using the SQLBuilder tool to write easily your SQL statements.

tInterbaseRow executes the SQL query stated onto the specified database. The Row suffix means the component implements a flow in the job design although it does not provide output.

**tInterbaseRow Standard properties**

These properties are used to configure tInterbaseRow running in the Standard Job framework.

The Standard tInterbaseRow component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Note: | When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to: |
| 1. | In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection. |
| 2. | In the child level, use a dedicated connection component to read that registered database connection. |
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Query type</strong></td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process.</td>
</tr>
</tbody>
</table>
for error-free rows. If needed, you can retrieve the rows on error via a Row > Rejects link.

**Advanced settings**

**Propagate QUERY’s recordset**
Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the use column list.

*Note:*
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of Object and this component is usually followed by tParseRecordSet.

**Use PreparedStatement**
Select this check box if you want to query the database using a PreparedStatement. In the Set PreparedStatement Parameter table, define the parameters represented by “?” in the SQL instruction of the Query field in the Basic Settings tab.

**Parameter Index:** Enter the parameter position in the SQL instruction.

**Parameter Type:** Enter the parameter type.

**Parameter Value:** Enter the parameter value.

*Note:*
This option is very useful if you need to execute the same query several times. Performance levels are increased.

**Commit every**
Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.

**tStat Catcher Statistics**
Select this check box to collect log data at the component level.

**Global Variables**

**Global Variables**

**QUERY:** the query statement being processed. This is a Flow variable and it returns a string.

**ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

**Usage rule**

This component offers the flexibility of the DB query and covers all possible SQL queries.

**Dynamic settings**

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Limitation**

This component requires installation of its related jar files.

**Related scenarios**

For related scenarios, see:

- **Scenario: Combining two flows for selective output** on page 2558
- For **tDBSQLRow** related scenario: see Procedure on page 625
- For **tMySQLRow** related scenario: see Scenario: Removing and regenerating a MySQL table index on page 2552.
tIntervalMatch

Returns a value based on a Join relation.

tIntervalMatch receives a main flow and aggregates it based on join to a lookup flow. Then it matches a specified value to a range of values and returns related information.

**tIntervalMatch Standard properties**

These properties are used to configure tIntervalMatch running in the Standard Job framework.

The Standard tIntervalMatch component belongs to the Data Quality family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>The schema already exists and is stored in the Repository, hence can be reused in various projects and job flowcharts. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Search Column**

Select the main flow column containing the values to be matched with a range of values

**Column (LOOKUP)**

Select the lookup flow column containing the values to be returned when the Join is ok.
Lookup Column (min) / Include the bound (min)

Select the column containing the minimum value of the range. Select the check box to include the minimum value of the range in the match.

Lookup Column (max) / Include the bound (max)

Select the column containing the maximum value of the range. Select the check box to include the maximum value of the range in the match.

Advanced settings

tStat Catcher Statistics

Select this check box to collect log data at the component level.

Global Variables

Global Variables

**NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

Usage

Usage rule

This component handles flow of data therefore it requires input and output, hence is defined as an intermediary step.

**Scenario: Identifying server locations based on their IP addresses**

This scenario describes a four-component Job that checks the server IP addresses listed in the main input file against a list of IP ranges given in a lookup file to identify the hosting country for each server.
Setting up the Job

About this task

The Job requires two `tFileInputDelimited` components, a `tIntervalMatch` component and a `tLogRow` component.

Procedure

1. Drop the components onto the design workspace.
2. Connect the components using Row > Main connection.
   Note that the connection from the second `tFileInputDelimited` component to the `tIntervalMatch` component will appear as a Lookup connection.

Configuring the components

Procedure

1. Double-click the first `tFileInputDelimited` component to open its Basic settings view.

2. Browse to the file to be used as the main input, which provides a list of servers and their IP addresses:

   Server;IP
   Server1;057.010.010.010
   Server2;001.010.010.100
   Server3;057.030.030.030
3. Click the [...] button next to Edit schema to open the [Schema] dialog box and define the input schema. According to the input file structure, the schema is made of two columns, respectively Server and IP, both of type String. Then click OK to close the dialog box.

4. Define the number of header rows to be skipped, and keep the other settings as they are.

5. Define the properties of the second tFileInputDelimited component similarly.

The file to be used as the input to the lookup flow in this example lists some IP address ranges and the corresponding countries:

StartIP;EndIP;Country
001.000.000.000;001.255.255.255;USA
002.006.190.056;002.006.190.063;UK
011.000.000.000;011.255.255.255;USA
057.000.000.000;057.255.255.255;France
012.063.178.060;012.063.178.063;Canada
053.000.000.000;053.255.255.255;Germany

Accordingly, the schema of the lookup flow should have the following structure:
6. Double-click the `tIntervalMatch` component to open its `Basic settings` view.

7. From the `Search Column` list, select the main flow column containing the values to be matched with the range values. In this example, we want to match the servers' IP addresses with the range values from the lookup flow.

8. From the `Column (LOOKUP)` list, select the lookup column that holds the values to be returned. In this example, we want to get the names of countries where the servers are hosted.

9. Set the min and max lookup columns corresponding to the range bounds defined in the lookup schema, `StartIP` and `EndIP` respectively in this example.

**Executing the Job**

**Procedure**

Press Ctrl+S to save your Job and press F6 to run it.

The name of the country where each server is hosted is displayed next to the IP address.

```
[statistics] connecting to socket on port 3452
[statistics] connected
Server1 | 057.010.010.010 | France
Server2 | 001.010.010.100 | USA
Server3 | 057.030.030.030 | France
Server4 | 053.010.010.100 | Germany
[statistics] disconnected
```
tIterateToFlow

Transforms non processable data into a processable flow. tIterateToFlow transforms a list into a data flow that can be processed.

tIterateToFlow Standard properties

These properties are used to configure tIterateToFlow running in the Standard Job framework.

The Standard tIterateToFlow component belongs to the Orchestration family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either Built-in or remote in the Repository. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
</table>
| | • View schema: choose this option to view the schema only.  
| | • Change to built-in property: choose this option to change the schema to Built-in for local changes.  
| | • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
| Built-in: The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide. |  |
| Repository: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job designs. Related topic: see Talend Studio User Guide. |  |

Mapping

<table>
<thead>
<tr>
<th>Column: Enter a name for the column to be created</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value: Press Ctrl+Space to access all of the available variables, be they global or user-defined.</td>
</tr>
</tbody>
</table>

Advanced Settings

| tStatCatcher Statistics | Select this check box to collect the log data at a component level. |
Global Variables

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

Usage

**Usage rule** This component is not startable (green background) and it requires an output component.

**Connections**

Outgoing links (from this component to another):

- **Row**: Main.
- **Trigger**: Run if; On Component Ok; On Component Error.

Incoming links (from one component to this one):

- **Row**: Iterate;

For further information regarding connections, see [Talend Studio User Guide](#).

Scenario: Transforming a list of files as data flow

The following scenario describes a Job that iterates on a list of files, picks up the filename and current date and transforms this into a flow, that gets displayed on the console.

- Drop the following components: **tFileList**, **tIterateToFlow** and **tLogRow** from the **Palette** to the design workspace.

- Connect the **tFileList** to the **tIterateToFlow** using an **iterate** link and connect the Job to the **tLogRow** using a **Row main** connection.

- In the **tFileList Component** view, set the directory where the list of files is stored.
In this example, the files are three simple .txt files held in one directory: Countries.

No need to care about the case, hence clear the Case sensitive check box.

Leave the Include Subdirectories check box unchecked.

Then select the tIterateToFlow component and click Edit Schema to set the new schema.

Add two new columns: Filename of String type and Date of date type. Make sure you define the correct pattern in Java.

Click OK to validate.

Notice that the newly created schema shows on the Mapping table.

In each cell of the Value field, press Ctrl+Space bar to access the list of global and user-specific variables.

For the Filename column, use the global variable: tFileList_1CURRENT_FILEPATH. It retrieves the current filepath in order to catch the name of each file, the Job iterates on.

For the Date column, use the Talend routine: TalendDate.getCurrentDate() (in Java).

Then on the tLogRow component view, select the Print values in cells of a table check box.

Save your Job and press F6 to execute it.
Starting job tIterateToFloW at 10:28 10/03/2010.

[statistics] connecting to socket on port 3773
[statistics] connected

<table>
<thead>
<tr>
<th>LogRow_1</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Filename</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>D:\Input\Countries\in-01.txt</td>
<td>2010-03-10 10:28:56</td>
</tr>
<tr>
<td>D:\Input\Countries\in-02.txt</td>
<td>2010-03-10 10:28:56</td>
</tr>
<tr>
<td>D:\Input\Countries\in-03.txt</td>
<td>2010-03-10 10:28:56</td>
</tr>
</tbody>
</table>

[statistics] disconnected
Job tIterateToFloW ended at 10:28 10/03/2010. [exit code=0]

The filepath displays on the Filename column and the current date displays on the Date column.
tJasperOutput

Creates a report in rich formats using Jaspersoft’s iReport.

This component is closely related to Jaspersoft’s report designer -- iReport. It reads and processes data from an input flow to create a report against a .jrxml report template defined via iReport.

tJasperOutput reads and processes data from an input flow to create a report against a .jrxml report template defined via iReport.

**tJasperOutput Standard properties**

These properties are used to configure tJasperOutput running in the Standard Job framework.

The Standard tJasperOutput component belongs to the Business Intelligence family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temp path</td>
<td>Path of temporary files.</td>
</tr>
<tr>
<td>Destination path</td>
<td>Path of the final report file.</td>
</tr>
<tr>
<td>File name/Stream</td>
<td>Name of the final report.</td>
</tr>
<tr>
<td>Report type</td>
<td>File type of the final report.</td>
</tr>
</tbody>
</table>

**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see the **Talend Studio User Guide**.
Repository: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Sync columns</th>
<th>Click to synchronize the output file schema with the input file schema. The Sync function only displays once the Row connection is linked with the output component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>iReport</td>
<td>Edit the command to provide the path of iReport’s execution file, e.g. replacing <strong>IREPORT_PATH</strong>\ with E:\Program Files\Jaspersoft\iReport-4.1.1\bin\ or giving the full path of the execution file such as &quot;E:\Program Files\Jaspersoft\iReport-4.1.1\bin\iReport.exe&quot;.</td>
</tr>
<tr>
<td>Launch</td>
<td>Click to run iReport.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify Locale</td>
<td>Select this check box to choose a locale from the Report Locale list.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>The first line of the Report Locale list is empty. You can click it to customize a locale.</td>
</tr>
<tr>
<td>Encoding</td>
<td>Select an encoding mode from this list. You can select Custom from the list to enter an encoding method in the field that appears.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
Usage

| Usage rule | This component is closely related to Jaspersoft’s report designer -- iReport. It reads and processes data from an input flow to create a report against a .jrxml report template defined via iReport. |

Scenario: Generating a report against a .jrxml template

The following Job reads data from a .csv file and creates a .pdf report based on an existing .jrxml report template. Note that the template file should be created via Jaspersoft’s iReport based on a file that shares the same schema with the source .csv file of this job.

Setting up the Job

Procedure

1. Drag and drop the following components from the Palette to the workspace: tFileInputDelimited and tJasperOutput.
2. Connect tFileInputDelimited and tJasperOutput using a Row link.

Configuring the input component

Procedure

1. Double-click the tFileInputDelimited component to display its Basic settings view.
2. Select Built-In from the Property Type drop-down list.

Note:
You can select **Repository** from the **Property Type** drop-down list to fill in the relevant fields automatically if the relevant metadata has been stored locally in the **Repository**. For more information about **Metadata**, see the **Talend Studio User Guide**.

3. Fill in the **File name/Stream** field to give the path and name of the source file, e.g. "C:/Documents and Settings/Andy ZHANG/nom.csv".

4. Keep the default settings for the **Row Separator** and **Field Separator** fields. You can also change them as needed.

5. Set 1 in the **Header** field and 0 in the **Footer** field. Leave the **Limit** field empty. You can also change them as needed.

6. Select **Built-In** from the **Schema** drop-down list and click **Edit schema** to define the data structure of the input file. In this case, the input file has 2 columns: **Nom** and **Prenom**.

Configuring the output component

**Procedure**

1. Double-click **tJasperOutput** to display its **Basic settings** view.

   ![Schema of FileInputDelimited_1](image)

   ![tJasperOutput_1](image)

   ![Configuring the output component](image)

   **Note:**
   
   The schema of the file, which is used to create a .jrxml template file via iReport, should be the same as that of the source file that is used to create the report.

2. Enter the full path of the report template file created via Jaspersoft's iReport in the **Jrxml file** field. You can click the three-dot button to browse.

3. Enter the path for the temporary files generated during the job execution in the **Temp path** field. You can click the three-dot button to browse.
4. Enter the path for the final report file generated during the job execution in the **Destination path** field. You can click the three-dot button to browse.

5. Enter the name for the final report file generated during the job execution in the **File name/Stream** field.

6. Select the format for the final report file generated during the job execution in the **Report type** field.

7. Click **Sync columns** to retrieve the schema from the previous component.

8. Enter the path of execution file of Jaspersoft’s iReport in the **iReport** field, e.g. replacing __IREPORT_PATH__ with `E:\Program Files\Jaspersoft\iReport-4.1.1\bin`. You can click the **Launch** button to run iReport.

   ![Note](image)

   This step is not mandatory. Yet, this helps you conveniently access the iReport software for relevant operations, e.g. creating a report template, etc.

**Job execution**

**Procedure**

1. Press **CTRL+S** to save your Job.

2. Press **F6** to execute it.

   You can find the file `out.pdf` in the folder specified in the **Destination path** field.
tJasperOutputExec

Creates a report in rich formats using Jaspersoft’s iReport and offers a performance gain as it functions as a combination of an input component and a tJasperOutput component.

This component is closely related to Jaspersoft’s report designer -- iReport. It reads and processes data from a source file to create a report against a .jrxml report template defined via iReport. tJasperOutputExec is used as a combination of an input component and a tJasperOutput component. The advantage of using two separate components is that data can be transformed before being used to generate a report and the input sources can be various and rich.

Reads and processes data from a source file to create a report against a .jrxml report template defined via iReport.

tJasperOutputExec Standard properties

These properties are used to configure tJasperOutputExec running in the Standard Job framework.

The Standard tJasperOutputExec component belongs to the Business Intelligence family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source file</td>
<td>Name of the source file.</td>
</tr>
<tr>
<td>Record delimiter</td>
<td>Delimiter of the records.</td>
</tr>
<tr>
<td>Destination path</td>
<td>Path of the final report file.</td>
</tr>
<tr>
<td>Use Default Output Name</td>
<td>Select this check box to use the default name for the report generated, which takes the source file's name.</td>
</tr>
<tr>
<td>Output Name</td>
<td>Name of the final report.</td>
</tr>
</tbody>
</table>

**Note:** This field does not appear if the Use Default Output Name box has been selected.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report type</td>
<td>File type of the final report.</td>
</tr>
<tr>
<td>iReport</td>
<td>Edit the command to provide the path of iReport’s execution file, e.g. replacing _IREPORT_PATH_ with E:\Program Files\Jaspersoft\iReport-4.1.1\bin\ or giving the full path of the execution file such as &quot;E:\Program Files\Jaspersoft\Report-4.1.1\bin\Report.exe&quot;.</td>
</tr>
<tr>
<td>Launch</td>
<td>Click to run iReport.</td>
</tr>
</tbody>
</table>
### Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
| **Specify Locale**              | Select this check box to choose a locale from the Report Locale list.  
Note: The first line of the Report Locale list is empty. You can click it to customize a locale. |
| **Encoding**                    | Select an encoding mode from this list. You can select Custom from the list to enter an encoding method in the field that appears.            |

### Global Variables

| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

### Usage

| Usage rule | This component is closely related to Jaspersoft’s report designer -- iReport. It reads and processes data from a source file to create a report against a .jrxml report template defined via iReport. |

### Related Scenario

For related scenarios, see **Scenario: Generating a report against a .jrxml template** on page 1829.
tJava

Extends the functionalities of a Talend Job using custom Java commands. tJava enables you to enter personalized code in order to integrate it in Talend program. You can execute this code only once.

**tJava Standard properties**

These properties are used to configure tJava running in the Standard Job framework. The Standard tJava component belongs to the Custom Code family. The component in this framework is generally available.

**Basic settings**

| Code | Type in the Java code you want to execute according to the task you need to perform. For further information about Java functions syntax specific to Talend, see Talend Studio Help Contents (Help > Developer Guide > API Reference).
|      | For a complete Java reference, check [http://docs.oracle.com/javaee/6/api/](http://docs.oracle.com/javaee/6/api/). |

**Advanced settings**

| Import | Enter the Java code to import, if necessary, external libraries used in the Code field of the Basic settings view. |

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

For a complete Java reference, check [http://docs.oracle.com/javaee/6/api/](http://docs.oracle.com/javaee/6/api/).
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is generally used as a one-component subjob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>You should know Java language.</td>
</tr>
</tbody>
</table>

Scenario: Printing out a variable content

The following scenario is a simple demo of the extended application of the tJava component. The Job aims at printing out the number of lines being processed using a Java command and the global variable provided in Talend Studio.

Setting up the Job

Procedure

1. Select and drop the following components from the Palette onto the design workspace: tFileInputDelimited, tFileOutputExcel, tJava.
2. Connect the tFileInputDelimited to the tFileOutputExcel using a Row Main connection. The content from a delimited txt file will be passed on through the connection to an xls-type of file without further transformation.
3. Then connect the tFileInputDelimited component to the tJava component using a Trigger > On Subjob Ok link. This link sets a sequence ordering tJava to be executed at the end of the main process.

Configuring the input component

Procedure

1. Set the Basic settings of the tFileInputDelimited component.
2. Define the path to the input file in the **File name** field.
   The input file used in this example is a simple text file made of two columns: **Names** and their respective **Emails**.

3. Click the **Edit Schema** button, and set the two-column schema. Then click **OK** to close the dialog box.

4. When prompted, click **OK** to accept the propagation, so that the **tFileOutputExcel** component gets automatically set with the input schema.

**Configuring the output component**

Set the output file to receive the input content without changes. If the file does not exist already, it will get created.
In this example, the Sheet name is Email and the Include Header box is selected.

**Configuring the tJava component**

**Procedure**

1. Then select the tJava component to set the Java command to execute.

2. In the Code area, type in the following command:

   ```java
   String var = "Nb of line processed: ";
   var = var + globalMap.get("tFileInputDelimited_1_NB_LINE");
   System.out.println(var);
   ```

   In this use case, we use the NB_LINE variable. To access the global variable list, press Ctrl + Space bar on your keyboard and select the relevant global parameter.

**Executing the Job**

**Procedure**

1. Press Ctrl+S to save your Job.
2. Press F6 to execute it.
Results

Nb of line processed: 4
Job JavaDb ended at 13:53 20/08/2007. [exit code=0]

The content gets passed on to the Excel file defined and the Number of lines processed are displayed on the Run console.
**tJavaDBInput**

Reads a database and extracts fields based on a query

tJavaDBInput executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Main row link.

**tJavaDBInput Standard properties**

These properties are used to configure tJavaDBInput running in the Standard Job framework.

The Standard tJavaDBInput component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Built-in: No property data stored centrally.</th>
<th>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Framework</th>
<th>Select your Java database framework on the list</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Database</th>
<th>Name of the database</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DB root path</th>
<th>Browse to your database root.</th>
</tr>
</thead>
</table>

| Username and Password | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |

| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. |

| Built-in: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide. |
Repository: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Query type** and **Query**
Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

**Advanced settings**

<table>
<thead>
<tr>
<th>Trim all the String/Char columns</th>
<th>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trim column</td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>QUERY</strong>: the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
For further information about variables, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Usage</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage rule</strong></td>
</tr>
<tr>
<td><strong>Limitation</strong></td>
</tr>
</tbody>
</table>

**Related scenarios**

For related topics, see:

See also the related topic in **tContextLoad**: Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
tJavaDBOutput

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tJavaDBOutput writes, updates, makes changes or suppresses entries in a database.

**tJavaDBOutput Standard properties**

These properties are used to configure tJavaDBOutput running in the Standard Job framework.
The Standard tJavaDBOutput component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

![Click to open database connection wizard](icon)

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see [Talend Studio User Guide](#).

**Framework**

Select your Java database framework on the list

**Database**

Name of the database

**DB root path**

Browse to your database root.

**Username and Password**

DB user authentication data.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**Table**

Name of the table to be written. Note that only one table can be written at a time

**Action on table**

On the table defined, you can perform one of the following operations:

- **None**: No operation is carried out.
- **Drop and create a table**: The table is removed and created again.
- **Create a table**: The table does not exist and gets created.
<table>
<thead>
<tr>
<th><strong>Action on data</strong></th>
<th><strong>Create a table if not exists</strong>: The table is created if it does not exist.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Drop table if exists and create</strong>: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear a table</strong>: The table content is deleted.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>On the data of the table defined, you can perform:</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, Job stops.</td>
<td></td>
</tr>
<tr>
<td><strong>Update</strong>: Make changes to existing entries</td>
<td></td>
</tr>
<tr>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
<td></td>
</tr>
<tr>
<td><strong>Update or insert</strong>: Update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
<td></td>
</tr>
<tr>
<td><strong>Delete</strong>: Remove entries corresponding to the input flow.</td>
<td></td>
</tr>
</tbody>
</table>

**Warning:**

*It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking Edit Schema and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the Advanced settings view where you can simultaneously define primary keys for the Update and Delete operations. To do that: Select the Use field options check box and then in the Key in update column, select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the Key in delete column for the Delete operation.*

<table>
<thead>
<tr>
<th><strong>Schema and Edit schema</strong></th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>

| **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. | When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com). |
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

**Die on error**

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

**Advanced settings**

<table>
<thead>
<tr>
<th>Commit every</th>
<th>Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional Columns</td>
<td>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</td>
</tr>
<tr>
<td><strong>Name</strong>: Type in the name of the schema column to be altered or inserted as new column</td>
<td><strong>SQL expression</strong>: Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td><strong>Position</strong>: Select <strong>Before</strong>, <strong>Replace</strong> or <strong>After</strong> following the action to be performed on the reference column</td>
<td><strong>Reference column</strong>: Type in a column of reference that the <strong>tDBOoutput</strong> can use to place or replace the new or altered column.</td>
</tr>
<tr>
<td>Use field options</td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td>Enable debug mode</td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>( \text{NB_LINE} ): the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \text{NB_LINE_UPDATED} ): the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>( \text{NB_LINE_INSERTED} ): the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>( \text{NB_LINE_DELETED} ): the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>( \text{NB_LINE_REJECTED} ): the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>( \text{ERROR_MESSAGE} ): the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

Usage rule

- This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.
- This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a Java database. It also allows you to create a reject flow using a Row > Rejekts link to filter data in error. For an example of tMysqlOutput in use, see Scenario: Retrieving data in error with a Reject link on page 2529.

Limitation

- Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Related scenarios

For related topics, see:
• Scenario: Inserting a column and altering data using tMysqlOutput on page 2521.
tJavaDBRow

Acts on the actual database structure or on the data (although without handling data) using the SQLBuilder tool to write easily your SQL statements.

tJavaDBRow executes the SQL query stated onto the specified database. The Row suffix means the component implements a flow in the job design although it doesn’t provide output.

tJavaDBRow Standard properties

These properties are used to configure tJavaDBRow running in the Standard Job framework.

The Standard tJavaDBRow component belongs to the Databases family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Framework</td>
<td>Select your Java database framework on the list</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>DB root path</td>
<td>Browse to your database root.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>Built-in: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td></td>
<td>Repository: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td></td>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
</tbody>
</table>
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Query type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: Fill in manually the query statement or build it graphically using SQLBuilder</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query</th>
<th>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Die on error</th>
<th>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</th>
</tr>
</thead>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>Propagate QUERY's recordset</th>
<th>Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the <strong>use column</strong> list.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Use PreparedStatement</th>
<th>Select this check box if you want to query the database using a PreparedStatement. In the <strong>Set PreparedStatement Parameter</strong> table, define the parameters represented by &quot;?&quot; in the SQL instruction of the <strong>Query</strong> field in the <strong>Basic Settings</strong> tab.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter Index</strong>: Enter the parameter position in the SQL instruction.</td>
<td></td>
</tr>
<tr>
<td><strong>Parameter Type</strong>: Enter the parameter type.</td>
<td></td>
</tr>
<tr>
<td><strong>Parameter Value</strong>: Enter the parameter value.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:**
This option is very useful if you need to execute the same query several times. Performance levels are increased

<table>
<thead>
<tr>
<th>Commit every</th>
<th>Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.</th>
</tr>
</thead>
</table>
Select this check box to collect log data at the component level.

Global Variables

**Global Variables**

**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**

This component offers the flexibility of the DB query and covers all possible SQL queries.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Related scenarios

For related topics, see:

- **Procedure** on page 625.
- **Scenario: Removing and regenerating a MySQL table index** on page 2552.
tJavaFlex

Provides a Java code editor that lets you enter personalized code in order to integrate it in Talend program.

tJavaFlex enables you to add Java code to the Start/Main/End code sections of this component itself. With tJavaFlex, you can enter the three java-code parts (start, main and end) that constitute a kind of component dedicated to do a desired operation.

**tJavaFlex Standard properties**

These properties are used to configure tJavaFlex running in the Standard Job framework.

The Standard tJavaFlex component belongs to the Custom Code family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Sync columns to retrieve the schema from the previous component in the Job.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In:</strong> You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository:</strong> You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.</td>
<td></td>
</tr>
<tr>
<td>You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
<td></td>
</tr>
<tr>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
<td></td>
</tr>
<tr>
<td>• <strong>View schema:</strong> choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Change to built-in property:</strong> choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Update repository connection:</strong> choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate</td>
<td></td>
</tr>
</tbody>
</table>
the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Data Auto Propagate**
Select this check box to automatically propagate the data to the component that follows.

**Start code**
Enter the Java code that will be called during the initialization phase.

**Main code**
Enter the Java code to be applied for each line in the data flow.

**End code**
Enter the Java code that will be called during the closing phase.

**Advanced settings**

**Import**
Enter the Java code that helps to import, if necessary, external libraries used in the Main code box of the Basic settings view.

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at a job level as well as at each component level.

**Global Variables**

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

**Usage**

**Usage rule**
You can use this component as a start, intermediate or output component. You can as well use it as a one-component subjob.

**Limitation**
You should know the Java language.
**Scenario 1: Generating data flow**

This scenario describes a two-components Job that generates a three-line data flow describing different personal titles (Miss, Mrs, and Mr) and displaying them on the console.

![Diagram of the Job](image)

**Setting up the Job**

**Procedure**

1. Drop **tJavaFlex** and **tLogRow** from the Palette onto the design workspace.
2. Connect the components together using a **Row > Main** link.

**Configuring the tJavaFlex component**

**Procedure**

1. Double-click **tJavaFlex** to display its **Basic settings** view and define its properties.

```
// Start part of your Java code
System.out.println("##
START\n##");
String [] valueArray = ("Miss", "Mrs", "Mr");

for (int i=0;i<valueArray.length;i++) {

```

2. Click the three-dot button next to **Edit schema** to open the corresponding dialog box where you can define the data structure to pass to the component that follows.
3. Click the [+] button to add two columns: **key** and **value** and then set their types to **Integer** and **String** respectively.

4. Click **OK** to validate your changes and close the dialog box.

5. In the **Basic settings** view of **tJavaFlex**, select the **Data Auto Propagate** check box to automatically propagate data to the component that follows.

   In this example, we do not want to do any transformation on the retrieved data.

6. In the **Start code** field, enter the code to be executed in the initialization phase.

   In this example, the code indicates the initialization of **tJavaFlex** by displaying the START message and sets up the loop and the variables to be used afterwards in the Java code:

   ```java
   System.out.println("## START
   #");
   String [] valueArray = {"Miss", "Mrs", "Mr"};
   for (int i=0;i<valueArray.length;i++) {
   }
   ```

7. In the **Main code** field, enter the code you want to apply on each of the data rows.

   In this example, we want to display each key with its value:

   ```java
   row1.key = i;
   row1.value = valueArray[i];
   ```

   **Warning:**

   *In the **Main code** field, "row1" corresponds to the name of the link that comes out of **tJavaFlex**. If you rename this link, you have to modify the code of this field accordingly.*

8. In the **End code** field, enter the code that will be executed in the closing phase.
In this example, the brace (curly bracket) closes the loop and the code indicates the end of the execution of `tJavaFlex` by displaying the END message:

```java
} System.out.println("## END");
```

9. If needed, double-click `tLogRow` and in its Basic settings view, click the [...] button next to Edit schema to make sure that the schema has been correctly propagated.

**Saving and executing the Job**

**Procedure**

1. Save your Job by pressing Ctrl+S.
2. Execute the Job by pressing F6 or clicking Run on the Run tab.

   Starting job tJavaFlex_scenario1 at 14:49 02/09/2009.

   ## START
   #
   0|Miss
   1|Mrs
   2|Mr
   #
   ## END
   Job tJavaFlex_scenario1 ended at 14:49 02/09/2009. [exit code=0]

The three personal titles are displayed on the console along with their corresponding keys.

**Scenario 2: Processing rows of data with tJavaFlex**

This scenario describes a two-component Job that generates random data and then collects that data and does some transformation on it line by line using Java code through the `tJavaFlex` component.

**Setting up the Job**

**Procedure**

1. Drop `tRowGenerator` and `tJavaFlex` from the Palette onto the design workspace.
2. Connect the components together using a **Row Main** link.

**Configuring the input component**

**Procedure**

1. Double-click **tRowGenerator** to display its **Basic settings** view and the [RowGenerator Editor] dialog box where you can define the component properties.

   ![RowGenerator Editor](image)

   **Schema**
   
<table>
<thead>
<tr>
<th>Column</th>
<th>Key</th>
<th>Type</th>
<th>N. Length</th>
<th>Functions</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td></td>
<td>int</td>
<td>3</td>
<td>...</td>
<td>1,2,3,4,5,6,7,8,9,10</td>
</tr>
<tr>
<td>date</td>
<td></td>
<td>String</td>
<td>20</td>
<td>...</td>
<td>&quot;2007-01-21&quot;, &quot;2007-02-22&quot;, &quot;2007-03-23&quot;</td>
</tr>
<tr>
<td>flag</td>
<td></td>
<td>boolean</td>
<td>10</td>
<td>...</td>
<td>true,false</td>
</tr>
</tbody>
</table>

   ![Column Editor](image)

   **Function parameters**
   
   ![Function parameters](image)

   2. Click the plus button to add four columns: **number**, **txt**, **date** and **flag**.

   3. Define the schema and set the parameters to the four columns according to the above capture.

   4. In the **Functions** column, select the three-dot function [...] for each of the defined columns.

   5. In the **Parameters** column, enter 10 different parameters for each of the defined columns. These 10 parameters corresponds to the data that will be randomly generated when executing **tRowGenerator**.

   6. Click **OK** to validate your changes and close the editor.

**Configuring the **tJavaFlex** component**

**Procedure**

1. Double-click **tJavaFlex** to display its **Basic settings** view and define the components properties.

   ![JavaFlex Editor](image)

   2. Click **Sync columns** to retrieve the schema from the preceding component.

   3. In the **Start code** field, enter the code to be executed in the initialization phase.

   ```java
   System.out.println("## START\n");
   int i = 0;
   ```
In this example, the code indicates the initialization of the **JavaFlex** component by displaying the START message and defining the variable to be used afterwards in the Java code:

```java
System.out.println("## START\n");
int i = 0;
```

4. In the **Main code** field, enter the code to be applied on each line of data.

In this example, we want to show the number of each line starting from 0 and then the number and the random text transformed to upper case and finally the random date set in the editor of **tRowGenerator**. Then, we create a condition to show if the status is **true** or **false** and we increment the number of the line:

```java
System.out.print(" row" + i + ":");
System.out.print(" # number:" + row1.number);
System.out.print (" | txt:" + row1.txt.toUpperCase());
System.out.print(" | date:" + row1.date);
if(row1.flag) System.out.println(" | flag: true");
else System.out.println(" | flag: false");
i++;
```

**Warning:**

*In the **Main code** field, "row1" corresponds to the name of the link that connects to **JavaFlex**. If you rename this link, you have to modify the code.*

5. In the **End code** field, enter the code that will be executed in the closing phase.

In this example, the code indicates the end of the execution of **JavaFlex** by displaying the END message:

```java
System.out.println("## END");
```
Saving and executing the Job

Procedure

1. Save your Job by pressing **Ctrl+S**.
2. Execute the Job by pressing **F6** or clicking **Run** on the **Run** tab.

```
Starting job tJavaFlex_scenario2 at 18:35 02/09/2009.

## START
#
rcw0 : number: 10 | txt: TEXT5 | date: 2006-05-25 | flag: false
rcw1 : number: 7  | txt: TEXT7 | date: 2006-06-26 | flag: true
rcw2 : number: 5  | txt: TEXT2 | date: 2006-05-25 | flag: false
rcw3 : number: 6  | txt: TEXT1 | date: 2005-08-28 | flag: true
rcw4 : number: 8  | txt: TEXT9 | date: 2006-05-25 | flag: false
rcw5 : number: 9  | txt: TEXT1 | date: 2007-01-21 | flag: false
rcw6 : number: 7  | txt: TEXT5 | date: 2004-11-21 | flag: true
rcw7 : number: 9  | txt: TEXT4 | date: 2005-08-29 | flag: true
rcw8 : number: 4  | txt: TEXT6 | date: 2006-06-26 | flag: false
rcw9 : number: 3  | txt: TEXT10 | date: 2006-05-25 | flag: false
#
## END

Job tJavaFlex_scenario2 ended at 18:35 02/09/2009. [exit code=0]
```

The console displays the randomly generated data that was modified by the java command set through **tJavaFlex**.
tJavaRow

Provides a code editor that lets you enter the Java code to be applied to each row of the flow. tJavaRow allows you to enter customized code which you can integrate in a Talend program.

**tJavaRow Standard properties**

These properties are used to configure tJavaRow running in the Standard Job framework. The Standard tJavaRow component belongs to the Custom Code family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Schema and Edit Schema</strong></th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.
Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

### Generate code
Click this button to automatically generate the code in the **Code** field to map the columns of the input schema with those of the output schema. This generation does not change anything in your schema.

The principle of this mapping is to relate the columns that have the same column name. Then you can adapt the generated code depending on the actual map you need.

### Code
Enter the Java code to be applied to each line of the data flow.

### Advanced settings

<table>
<thead>
<tr>
<th>Import</th>
<th>Enter the Java code to import, if necessary, external libraries used in the <strong>Code</strong> field of the <strong>Basic settings</strong> view.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect the log data at a component level.</td>
</tr>
</tbody>
</table>

### Global Variables

| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see **Talend Studio User Guide**. To enter a global variable (for example **COUNT** of **tFileRowCount**) in the **Code** box, you need to type in the entire piece of code manually, that is to say `((Integer)globalMap.get("tFileRowCount_COUNT")).` |

### Usage

| Usage rule | This component is used as an intermediary between two other components. It must be linked to both an input and an output component. |
| Function | \texttt{tJavaRow} allows you to enter customized code which you can integrate in a Talend programme. With \texttt{tJavaRow}, you can enter the Java code to be applied to each row of the flow. |
| Purpose | \texttt{tJavaRow} allows you to broaden the functionality of Talend Jobs, using the Java language. |
| Limitation | Knowledge of Java language is necessary. |

**Scenario: Transforming data line by line using \texttt{tJavaRow}**

In this scenario, the information of a few cities read from an input delimited file is transformed using Java code through the \texttt{tJavaRow} component and printed on the console.

**Setting up the Job**

**Procedure**

1. Drop a \texttt{tFileInputDelimited} component and a \texttt{tJavaRow} component from the \texttt{Palette} onto the design workspace, and label them to better identify their roles in the Job.
2. Connect the two components using a \texttt{Row > Main} connection.

**Configuring the components**

**Procedure**

1. Double-click the \texttt{tFileInputDelimited} component to display its \texttt{Basic settings} view in the \texttt{Component} tab.
2. In the File name/Stream field, type in the path to the input file in double quotation marks, or browse to the path by clicking the [...] button, and define the first line of the file as the header. In this example, the input file has the following content:

<table>
<thead>
<tr>
<th>City</th>
<th>Population</th>
<th>LandArea</th>
<th>PopDensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beijing</td>
<td>10233000</td>
<td>1418</td>
<td>7620</td>
</tr>
<tr>
<td>Moscow</td>
<td>10452000</td>
<td>1081</td>
<td>9644</td>
</tr>
<tr>
<td>Seoul</td>
<td>10422000</td>
<td>605</td>
<td>17215</td>
</tr>
<tr>
<td>Tokyo</td>
<td>8731000</td>
<td>617</td>
<td>14151</td>
</tr>
<tr>
<td>New York</td>
<td>8310000</td>
<td>789</td>
<td>10452</td>
</tr>
</tbody>
</table>

3. Click the [...] button next to Edit schema to open the [Schema] dialog box, and define the data structure of the input file. Then, click OK to validate the schema setting and close the dialog box.

4. Double-click the tJavaRow component to display its Basic settings view in the Component tab.

5. Click Sync columns to make sure that the schema is correctly retrieved from the preceding component.

6. In the Code field, enter the code to be applied on each line of data based on the defined schema columns.

In this example, we want to transform the city names to upper case, group digits of numbers larger than 1000 using the thousands separator for ease of reading, and print the data on the console:

```java
System.out.print("\n" + input_row.City.toUpperCase() + ":");
System.out.print(" - Population: " + FormatterUtils.format_Number(String.valueOf(input_row.Population), ',', '') + " people");
System.out.print(" - Land area: " + FormatterUtils.format_Number(String.valueOf(input_row.LandArea), ',', '') + " km2");
System.out.print(" - Population density: " + FormatterUtils.format_Number(String.valueOf(input_row.PopDensity), ',', '.') + " people/km2");
```
System.out.print("\n - Population density: "+ FormatterUtils.format_Number(String.valueOf(input_row.PopDensity), ',', '.') + " people/km2\n");

Note:
In the Code field, input_row refers to the link that connects to tJavaRow.

Saving and executing the Job

Procedure
1. Press Ctrl+S to save your Job.
2. Press F6 or click Run on the Run tab to execute the Job.

The city information is transformed by the Java code set through tJavaRow and displayed on the console.
tJDBCClose

Closes an active JDBC connection to release the occupied resources.

**tJDBCClose Standard properties**

These properties are used to configure tJDBCClose running in the Standard Job framework.

The Standard tJDBCClose component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Component</td>
<td>Select the component that opens the connection you need to close from the drop-down list.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

**Usage**

| Usage rule | This component is to be used along with JDBC components, especially with **tJDBCConnection** and **tJDBCCommit**. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, |
when your Job has to be deployed and executed independent of Talend Studio.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

## Related scenarios

No scenario is available for the Standard version of this component yet.
tJDBCColumnList

Lists all column names of a given JDBC table. tJDBCColumnList iterates on all columns of a given table through a defined JDBC connection.

**tJDBCColumnList Standard properties**

These properties are used to configure tJDBCColumnList running in the Standard Job framework. The Standard tJDBCColumnList component belongs to the Databases family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Select the type of the database to be accessed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tJDBCConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Table name</td>
<td>Enter the name of the table.</td>
</tr>
<tr>
<td>Die on error</td>
<td>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row &gt; Reject link.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics     | Select this check box to collect log data at the component level. |

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>CURRENT_COLUMN: the name of the column currently iterated upon. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CURRENT_COLUMN_TYPE: the ID of the type of the column currently iterated upon. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>CURRENT_COLUMN_TYPE_NAME: the name of the type of the column currently iterated upon. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>CURRENT_COLUMN_PRECISION: the precision of the column currently iterated upon. This is a Flow variable, and it returns an integer.</td>
</tr>
</tbody>
</table>
**CURRENT_COLUMN_SCALE**: the scale of the column currently iterated upon. This is a Flow variable, and it returns an integer.

**NB_COLUMN**: the number of columns iterated upon so far. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

**Usage rule**

This component is to be used along with JDBC components, especially with **tJDBCConnection**.

### Related scenario

For **tJDBCColumnList** related scenario, see Scenario: Iterating on a DB table and listing its column names on page 2470.
**tJDBCCommit**

Commits in one go a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

`tJDBCCommit` validates the data processed through the Job into the connected DB.

**`tJDBCCommit` Standard properties**

These properties are used to configure `tJDBCCommit` running in the Standard Job framework.

The `Standard tJDBCCommit` component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Component</td>
<td>Select the component that opens the database connection to be reused by this component.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Select this check box to close the database connection once the component has performed its task. Clear this check box to continue to use the selected connection once the component has performed its task. If this component is linked to your Job via a Row &gt; Main connection, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of the first row commit.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| `tStatCatcher Statistics` | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| `ERROR_MESSAGE` | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other tJDBC* components, especially with the tJDBCConnection and tJDBCRollback components.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Related scenario

For tJDBCCommit related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476.
tJDBCConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

tJDBCConnection Standard properties

These properties are used to configure tJDBCConnection running in the Standard Job framework.
The Standard tJDBCConnection component belongs to the Databases and the ELT families.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
</table>
| Property Type | Select the way the connection details will be set.  
• **Built-In**: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.  
• **Repository**: The connection details stored centrally in Repository > Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in. |
| JDBC URL | The JDBC URL of the database to be used. For example, the JDBC URL for the Amazon Redshift database is jdbc:redshift://endpoint:port/database. |
| Drivers | Complete this table to load the driver JARs needed. To do this, click the [+] button under the table to add as many rows as needed, each row for a driver JAR, then select the cell and click the [...] button at the right side of the cell to open the Module wizard from which you can select the driver JAR of your interest. For example, the driver jar RedshiftJDBC41-1.1.13.1013.jar for the Redshift database. |
| Driver Class | The class name for the specified driver between double quotation marks. For example, for the RedshiftJDBC41-1.1.13.1013.jar driver, the name to be entered is com.amazon.redshift.jdbc41.Driver. |
| **Use Id and Password** | The database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Use or register a shared DB Connection** | Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the **Shared DB Connection Name** field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child. This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the **tRunJob** component. Using a shared connection together with a **tRunJob** component with either of these two options enabled will cause your Job to fail. This check box is not available when the **Specify a data source alias** check box is selected. |
| **Specify a data source alias** | Select this check box and in the **Data source alias** field displayed, specify the alias of a data source created on Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503. This check box is not available when the **Use or register a shared DB Connection** check box is selected. |

**Advanced settings**

| **Use Auto-Commit** | Select this check box to activate the auto commit mode. |
| **Auto Commit** | Select this check box to commit any changes to the database automatically upon the transaction. With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component. Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component. |
tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Global Variables

ERROR_MESSAGE

The error message generated by the component when an error occurs. This is an After variable and it returns a string.

Usage

Usage rule

This component is more commonly used with other tJDBC* components, especially with the tDBCCommit and tDBCRollback components.

Related scenario

For tDBCCConnection related scenario, see tMysqlConnection on page 2475
tJDBCInput

Reads any database using a JDBC API connection and extracts fields based on a query.

tJDBCInput executes a database query with a strictly defined order which must correspond to the
schema definition. Then it passes on the field list to the next component via a Main row link.

**tJDBCInput Standard properties**

These properties are used to configure tJDBCInput running in the Standard Job framework.
The Standard tJDBCInput component belongs to the Databases family.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related
to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components
on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td>• Built-In: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
<td></td>
</tr>
<tr>
<td>• Repository: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
<td></td>
</tr>
</tbody>
</table>

This property is not available when other connection component is selected from the Connection Component drop-down list.

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

| Connection Component | Select the component that opens the database connection to be reused by this component. |
### JDBC URL

The JDBC URL of the database to be used. For example, the JDBC URL for the Amazon Redshift database is `jdbc:redshift://endpoint:port/database`.

### Drivers

Complete this table to load the driver JARs needed. To do this, click the `[+]` button under the table to add as many rows as needed, each row for a driver JAR, then select the cell and click the `[...]` button at the right side of the cell to open the Module wizard from which you can select the driver JAR of your interest. For example, the driver jar `RedshiftJDBC41-1.1.13.1013.jar` for the Redshift database.

### Driver Class

The class name for the specified driver between double quotation marks. For example, for the `RedshiftJDBC41-1.1.13.1013.jar` driver, the name to be entered is `com.amazon.redshift.jdbc41.Driver`.

### Use Id and Password

The database user authentication data.

To enter the password, click the `[...]` button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

Click Edit schema to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

### Table Name

The name of the table from which data will be retrieved.
**Query Type** and **Query**

Specify the database query statement paying particularly attention to the properly sequence of the fields which must correspond to the schema definition.

- **Built-In**: Fill in the query statement in the **Query** field manually or click the [...] button next to the **Query** field to build the statement graphically using the SQLBuilder.
- **Repository**: Select the relevant query stored in the Repository by clicking the [...] button next to it and in the pop-up **Repository Content** dialog box, select the query to be used, and the **Query** field will be automatically filled in.

**Guess Query**

Click this button to generate query in the **Query** field based on the defined table and schema.

**Guess Schema**

Click this button to generate schema columns based on the query defined in the **Query** field.

**Specify a data source alias**

Select this check box and in the **Data source alias** field displayed, specify the alias of a data source created on Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

This property is not available when other connection component is selected from the **Connection Component** drop-down list.

**Advanced settings**

**Use cursor**

Select this check box to specify the number of rows you want to work with at any given time. This option optimises performance.

**Trim all the String/Char columns**

Select this check box to remove leading whitespace and trailing whitespace from all String/Char columns.

**Check column to trim**

Select the check box for corresponding column to remove leading whitespace and trailing whitespace from it.

This property is not available when the **Trim all the String/Char columns** check box is selected.

**Enable Mapping File for Dynamic**

Select this check box to use the specified metadata mapping file when reading data from a dynamic type column. This check box is cleared by default.

With this check box selected, you can specify the metadata mapping file to use by selecting a type of database from the **Mapping File** drop-down list.
For more information about metadata mapping files, see the section on type conversion of Talend Studio User Guide.

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>NB_LINE</td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>QUERY</td>
<td>The query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This component covers all possible SQL queries for any database using a JDBC connection.</td>
</tr>
</tbody>
</table>

**Dynamic settings**
Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Related scenarios

For related topics, see:

Related topic in **tContextLoad**: see Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
tJDBCOutput

Executes the action defined on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tJDBCOutput writes, updates, makes changes or suppresses entries in any type of database connected to a JDBC API.

**tJDBCOutput Standard properties**

These properties are used to configure tJDBCOutput running in the Standard Job framework.

The Standard tJDBCOutput component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td>• Built-In: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
<td></td>
</tr>
<tr>
<td>• Repository: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
<td></td>
</tr>
</tbody>
</table>

This property is not available when other connection component is selected from the Connection Component drop-down list.

| Connection Component | Select the component that opens the database connection to be reused by this component. |

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.
<p>| <strong>JDBC URL</strong> | The JDBC URL of the database to be used. For example, the JDBC URL for the Amazon Redshift database is <code>jdbc:redshift://endpoint:port/database</code>. |
| <strong>Drivers</strong> | Complete this table to load the driver JARs needed. To do this, click the <code>[+]</code> button under the table to add as many rows as needed, each row for a driver JAR, then select the cell and click the <code>[...]</code> button at the right side of the cell to open the Module wizard from which you can select the driver JAR of your interest. For example, the driver jar <code>RedshiftJDBC41-1.1.13.1013.jar</code> for the Redshift database. |
| <strong>Driver Class</strong> | The class name for the specified driver between double quotation marks. For example, for the <code>RedshiftJDBC41-1.1.13.1013.jar</code> driver, the name to be entered is <code>com.amazon.redshift.jdbc41.Driver</code>. |
| <strong>Use Id and Password</strong> | The database user authentication data. To enter the password, click the <code>[...]</code> button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <code>OK</code> to save the settings. |
| <strong>Table Name</strong> | The name of the table into which data will be written. |
| <strong>Data Action</strong> | Select an action to be performed on data of the table defined. |
| | • <strong>Insert</strong>: Add new entries to the table. If duplicates are found, job stops. |
| | • <strong>Update</strong>: Make changes to existing entries. |
| | • <strong>Insert or update</strong>: Insert a new record in the index pool. If the record with the given reference already exists, an update would be made. |
| | • <strong>Update or insert</strong>: Update the record with the given reference. If the record does not exist in the index pool, a new record would be inserted. |
| | • <strong>Delete</strong>: Remove entries corresponding to the input flow. |
| <strong>Warning</strong>: | It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking Edit Schema and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the Advanced settings view where you can simultaneously define primary keys for the Update and Delete operations. To do that: Select the Use field options check box and then in the Key in update column, select the check boxes next to the column names you want to use as a base for the Update operation. |</p>
<table>
<thead>
<tr>
<th>Clear data in table</th>
<th>Select this check box to clear data in the table before performing the action defined.</th>
</tr>
</thead>
</table>
| **Schema** and **Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.  
  - **Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](https://help.talend.com).  
  - **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](https://help.talend.com).  
  
  When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.  
  
  You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).  
  
  Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.  
  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. |
| **Guess Schema** | Click this button to generate schema columns based on the settings of database table columns. |
| **Die on error** | Select the check box to stop the execution of the Job when an error occurs.  
  
  Clear the check box to skip any rows on error and complete the process for error-free rows.  
  
  When errors are skipped, you can collect the rows on error using a **Row > Reject** connection. |
| **Specify a data source alias** | Select this check box and in the **Data source alias** field displayed, specify the alias of a data source created on Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend |
Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

If you use the component’s own DB configuration, your data source connection will be closed at the end of the component. To prevent this from happening, use a shared DB connection with the data source alias specified.

This property is not available when other connection component is selected from the Connection Component drop-down list.

### Advanced settings

<table>
<thead>
<tr>
<th>Commit every</th>
<th>Specify the number of rows to be processed before committing batches of rows together into the database. This option ensures transaction quality (but not rollback) and, above all, better performance at executions.</th>
</tr>
</thead>
</table>
| Additional Columns | This option allows you to call SQL functions to perform actions on columns, which are not insert, update or delete actions, or actions that require particular preprocessing. It is not offered if you create (with or without drop) the database table.  
  1. **Name**: The name of the schema column to be inserted, or the name of the schema column used to replace an existing column.  
  2. **SQL expression**: The SQL statement to be executed in order to insert or replace relevant column.  
  3. **Position**: Select Before, After, or Replace according to the action to be performed on the reference column.  
  4. **Reference column**: The name of the reference column that can be used to locate the new column to be inserted or that will be replaced. |
| Use field options | Select this check box and in the Fields options table displayed, select the check box for the corresponding column to customize a request, particularly if multiple actions are being carried out on the data.  
  1. **Key in update**: Select the check box for the corresponding column based on which data is updated.  
  2. **Key in delete**: Select the check box for the corresponding column based on which data is deleted.  
  3. **Updatable**: Select the check box if data in the corresponding column can be updated.  
  4. **Insertable**: Select the check box if data in the corresponding column can be inserted. |
Enable debug mode
Select this check box to display each step during processing entries in a database.

Use Batch
Select this check box to activate the batch mode for data processing, and in the Batch Size field displayed, specify the number of records to be processed in each batch.

tStatCatcher Statistics
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Enable parallel execution
Select this check box to perform high-speed data processing by treating multiple data flows simultaneously. This feature depends on the database or the application ability to handle multiple inserts in parallel as well as the number of CPU affected. With this check box selected, you need to specify the number of parallel executions desired in the Number of parallel executions field displayed.

Note: When parallel execution is enabled, it is not possible to use global variables to retrieve return values.

Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>NB_LINE</td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_INSERTED</td>
<td>The number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_UPDATED</td>
<td>The number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_DELETED</td>
<td>The number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_REJECTED</td>
<td>The number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>QUERY</td>
<td>The query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

Usage rule
This component offers the flexibility benefit of the database query and covers all of the SQL queries possible.

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a JDBC database. It also allows you to
create a reject flow using a **Row > Rejects** link to filter data in error. For an example of **tMySqlOutput** in use, see **Scenario: Retrieving data in error with a Reject link** on page 2529.

<table>
<thead>
<tr>
<th>Dynamic settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click the [+ button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. For examples on using dynamic parameters, see <strong>Scenario: Reading data from databases through context-based dynamic connections</strong> on page 2497 and <strong>Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters</strong> on page 489. For more information on <strong>Dynamic settings</strong> and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

**Related scenarios**

For **tJDBCOutput** related topics, see:

- **Scenario: Inserting a column and altering data using **tMysqlOutput** on page 2521.**
tJDBCRollback

Avoids committing part of a transaction accidentally by canceling the transaction committed in the connected database.

**tJDBCRollback Standard properties**

These properties are used to configure tJDBCRollback running in the Standard Job framework. The Standard tJDBCRollback component belongs to the Databases family.

The component in this framework is generally available.

![Note:](image)

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Component</td>
<td>Select the component that opens the database connection to be reused by this component.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Select this check box to close the database connection once the component has performed its task. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

**Usage**

| Usage rule | This component is more commonly used with other tJDBC* components, especially with the tJDBCConnection and tJDBCCommit components. |
| Dynamic settings | Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. |
This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenario

For tJDBCRollback related scenario, see tMysqlRollback on page 2546
tJDBCRow

Acts on the actual DB structure or on the data (although without handling data) using the SQLBuilder tool to write easily your SQL statements.

tJDBCRow is the component for any type database using a JDBC API. It executes the SQL query stated onto the specified database. The row suffix means the component implements a flow in the job design although it doesn’t provide output.

tJDBCRow Standard properties

These properties are used to configure tJDBCRow running in the Standard Job framework.

The Standard tJDBCRow component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td>• Built-In: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
<td></td>
</tr>
<tr>
<td>• Repository: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
<td></td>
</tr>
<tr>
<td>Connection Component</td>
<td>Select the component that opens the database connection to be reused by this component.</td>
</tr>
<tr>
<td>JDBC URL</td>
<td>The JDBC URL of the database to be used. For example, the JDBC URL for the Amazon Redshift database is jdbc:redshift://endpoint:port/database.</td>
</tr>
<tr>
<td>Drivers</td>
<td>Complete this table to load the driver JARs needed. To do this, click the [+] button under the table to add as many rows as needed, each row for a driver JAR, then select the cell and click the [...] button at the right side...</td>
</tr>
</tbody>
</table>
of the cell to open the **Module** wizard from which you can select the driver JAR of your interest. For example, the driver JAR `RedshiftJDBC41-1.1.13.1013.jar` for the Redshift database.

<table>
<thead>
<tr>
<th><strong>Driver Class</strong></th>
<th>The class name for the specified driver between double quotation marks. For example, for the <code>RedshiftJDBC41-1.1.13.1013.jar</code> driver, the name to be entered is <code>com.amazon.redshift.jdbc41.Driver</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Driver Class</strong></td>
<td>The class name for the specified driver between double quotation marks. For example, for the <code>RedshiftJDBC41-1.1.13.1013.jar</code> driver, the name to be entered is <code>com.amazon.redshift.jdbc41.Driver</code>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Use Id and Password</strong></th>
<th>The database user authentication data. To enter the password, click the <code>[...]</code> button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</th>
</tr>
</thead>
</table>

| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.  
  - **Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.  
  - **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.  
  Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

<table>
<thead>
<tr>
<th><strong>Table Name</strong></th>
<th>The name of the table to be processed.</th>
</tr>
</thead>
</table>

| **Query Type and Query** | Specify the database query statement paying particularly attention to the properly sequence of the fields which must correspond to the schema definition.  
  - **Built-In**: Fill in the query statement in the **Query** field manually or click the `[...]` button next to the **Query** field to build the statement graphically using the SQLBuilder.  
  - **Repository**: Select the relevant query stored in the Repository by clicking the `[...]` button next to it and |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

1885
in the pop-up **Repository Content** dialog box, select the query to be used, and the **Query** field will be automatically filled in.

<table>
<thead>
<tr>
<th><strong>Guess Query</strong></th>
<th>Click this button to generate query in the <strong>Query</strong> field based on the defined table and schema.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Specify a data source alias</strong></td>
<td>Select this check box and in the <strong>Data source alias</strong> field displayed, specify the alias of a data source created on Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503. If you use the component’s own DB configuration, your data source connection will be closed at the end of the component. To prevent this from happening, use a shared DB connection with the data source alias specified. This property is not available when other connection component is selected from the <strong>Connection Component</strong> drop-down list.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a <strong>Row &gt; Reject</strong> connection.</td>
</tr>
<tr>
<td><strong>Advanced settings</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Propagate QUERY’s recordset</strong></td>
<td>Select this check box to propagate the result of the query to the output flow. From the <strong>use column</strong> list displayed, you need to select a column into which the query result will be inserted. This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the query’s recordset should be set to the Object type and this component is usually followed by a <strong>tParseRecordSet</strong> component.</td>
</tr>
</tbody>
</table>
| **Use PreparedStatement** | Select this check box if you want to query the database using a prepared statement. In the **Set PreparedStatement Parameters** table displayed, specify the value for each parameter represented by a question mark (?) in the SQL statement defined in the **Query** field.  
  - **Parameter Index**: the position of the parameter in the SQL statement.  
  - **Parameter Type**: the data type of the parameter.  
  - **Parameter Value**: the value of the parameter. |
For a related use case of this property, see Scenario: Using PreparedStatement objects to query data on page 2553.

**Commit every**
Specify the number of rows to be processed before committing batches of rows together into the database. This option ensures transaction quality (but not rollback) and, above all, better performance at executions.

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUERY</td>
<td>The query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
</tbody>
</table>

### Usage

**Usage rule**
This component offers the flexibility of the DB query for any database using a JDBC connection and covers all possible SQL queries.

**Dynamic settings**
Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Related scenarios

For related topics, see:
- Scenario: Combining two flows for selective output on page 2558.
- Procedure on page 625.
- Scenario: Removing and regenerating a MySQL table index on page 2552.
**tJDBCSCDELT**

Tracks data changes in a source database table using SCD (Slowly Changing Dimensions) Type 1 method and/or Type 2 method and writes both the current and historical data into a specified SCD dimension table.

**tJDBCSCDELT Standard properties**

These properties are used to configure tJDBCSCDELT running in the Standard Job framework. The Standard tJDBCSCDELT component belongs to two families: Business Intelligence and Databases. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

**Use an existing connection**  
Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**JDBC URL**  
The JDBC URL of the database to be used. For example, the JDBC URL for the Amazon Redshift database is `jdbc:redshift://endpoint:port/database`.

**Driver JAR**  
Complete this table to load the driver JARs needed. To do this, click the `[+]` button under the table to add as many rows as needed, each row for a driver JAR, then select the cell and click the [...] button at the right side of the cell to open the Select Module wizard from which you can select the driver JAR of your interest. For example, the driver jar `RedshiftJDBC41-1.1.13.1013.jar` for the Redshift database.

**Driver Class**  
Enter the class name for the specified driver between double quotation marks. For example, for the `RedshiftJDBC41-1.1.13.1013.jar` driver, the name to be entered is `com.amazon.redshift.jdbc41.Driver`.

**Username and Password**  
The database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter
the password between double quotes and click **OK** to save the settings.

<table>
<thead>
<tr>
<th><strong>DB Schema</strong></th>
<th>Specify the name of the database schema. For the database that does not support schema, just leave this field empty.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source table</strong></td>
<td>Specify the name of the source input table whose data changes will be captured by SCD.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Specify the name of the dimension table into which the data captured by SCD will be written.</td>
</tr>
</tbody>
</table>
| **Action on table** | Select an operation to be performed on the table defined.  
  - **None**: No operation is carried out.  
  - **Drop and create table**: The table is removed and created again.  
  - **Create table**: The table does not exist and gets created.  
  - **Create table if does not exist**: The table is created if it does not exist.  
  - **Drop table if exist and create**: The table is removed if it already exists and created again.  
  - **Clear table**: The table content is deleted. You have the possibility to rollback the operation.  
  - **Truncate table**: The table content is deleted. You do not have the possibility to rollback the operation. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields.  
  - **Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](#).  
  - **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](#).  
  Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No**. |
upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th><strong>Surrogate key</strong></th>
<th>Set the column where the generated surrogate key will be stored. A surrogate key can be generated based on a method selected on the <strong>Creation</strong> list.</th>
</tr>
</thead>
</table>
| **Creation**       | Select any of the following methods to be used for generating the surrogate key.  
|                    | • **Auto increment**: an auto-incrementing integer will be used.  
|                    | • **DB sequence**: a sequence will be used. |
| **Source keys**    | Specify one or more columns used as the key(s) that ensure the unicity of the incoming data. |
| **Use SCD type 1 fields** | Select this check box and in the **SCD type 1 fields** table displayed, specify the column(s) whose value changes will be tracked using Type 1 SCD. |
| **SCD type 1 fields** | Click the [+ ] button to add as many rows as needed, each row for a column, then click the right side of the cell and from the drop-down list displayed select the column whose value changes will be tracked using Type 1 SCD. |
| **Use SCD type 2 fields** | Select this check box and in the **SCD type 2 fields** table displayed, specify the column(s) whose value changes will be tracked using Type 2 SCD. |
| **SCD type 2 fields** | Click the [+ ] button to add as many rows as needed, each row for a column, then click the right side of the cell and from the drop-down list displayed select the column whose value changes will be tracked using Type 2 SCD. |
| **Start date**     | Specify the column that holds the start date for type 2 SCD. |
| **End date**       | Specify the column that holds the end date for type 2 SCD. |
| **Log active status** | Select this check box and from the **Active field** drop-down list displayed, select the column that holds the true or false status value, which helps to spot the active record for type 2 SCD. |
| **Log versions**   | Select this check box and from the **Version field** drop-down list displayed, select the column that holds the version number of the record for type 2 SCD. |
| **Mapping**        | Specify the metadata mapping file for the database to be used. The metadata mapping file is used for the data type conversion between database and Java. For more information about the metadata mapping, see the related documentation for Type mapping. |
Advanced settings

<table>
<thead>
<tr>
<th>Source fields value include Null</th>
<th>Select this check box to allow the source columns to have Null values. The source columns here refer to the fields defined in the SCD type 1 fields and SCD type 2 fields tables.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug mode</td>
<td>Select this check box to display each step during processing the data in a database.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

Global Variables

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as a standalone component of a Job or Subjob.</th>
</tr>
</thead>
</table>

Tracking data changes in a Snowflake table using the tJDBCSCDELT component

This scenario describes a Job that captures the employee data changes in a Snowflake table using SCD (Slowly Changing Dimensions) Type 1 and Type 2 methods implemented by the tJDBCSCDELT component, and writes both the current and historical data in a SCD dimension table.

The input data contains various employee details including name, role, salary, and another id column is added to help ensuring the unicity of the input data.

At first, the following employee data is inserted to a new Snowflake table.

```
#id;name;role;salary
111;Mark Smith;tester;15000.00
222;Thomas Johnson;developer;18000.00
333;Teddy Brown;tester;16000.00
```

Later, the table is updated with the following renewed employee data.

```
#id;name;role;salary
111;Mark Smith;tester;15000.00
222;Thomas Johnson;tester;18000.00
333;Teddy Brown;writer;17000.00
444;John Clinton;developer;19000.00
```

You can see the role of Thomas Johnson is changed from developer to tester, the role of Teddy Brown is changed from tester to writer, and his salary is raised from 16000.00 to 17000.00. Besides, a new employee record with id 444 is inserted. In this scenario,
• the existing name and role data will be overwritten by the new data, so SCD Type 1 method will be performed on them, and

• the full history of the salary data will be retained, and a new record with the changed data will be always created and the previous record will be closed, so SCD Type 2 method will be performed on it.

For more information about SCD types, see SCD management methodology on page 2566.

**Creating a Job for tracking data changes in a Snowflake table using tJDBCSCDELT**

**Procedure**

1. Create a new Job and add a tJDBCConnection component, two tJDBCRow components, two tFixedFlowInput components, two tJDBCOutput components, two tJDBCSCDELT components, two tJDBCInput components, and two tLogRow components to the Job.
2. Link the first `tFixedFlowInput` component to the first `tJDBCOutput` component using a Row > Main connection.

3. Do the same to link the first `tJDBCInput` component to the first `tLogRow` component, the second `tFixedFlowInput` component to the second `tJDBCOutput` component, and the second `tJDBCInput` component to the second `tLogRow` component.

4. Link the `tJDBCConnection` component to the first `tJDBCRow` component using a Trigger > On Subjob Ok connection.

5. Do the same to link the first `tJDBCRow` component to the second `tJDBCRow` component, the second `tJDBCRow` component to the first `tFixedFlowInput` component, the first `tFixedFlowInput` component to the first `tJDBCSCDELT` component, the first `tJDBCSCDELT` component to the first `tJDBCInput` component, the first `tJDBCInput` component to the second `tFixedFlowInput` component, the second `tFixedFlowInput` component to the second `tJDBCSCDELT` component, the second `tJDBCSCDELT` component to the second `tJDBCInput` component.

Opening a connection to a Snowflake database

Procedure

1. Double-click the `tJDBCConnection` component to open its Basic settings view.

2. In the JDBC URL field, enter the connection string to connect to Snowflake using the JDBC driver.

   For more information about how to use the JDBC driver to connect to Snowflake and how to specify the connection string, see JDBC Driver Connection String.

3. Click the [+] button below the Driver JAR table to add a line, and in the Jar name cell of the new line, enter the name of the jar file for the Snowflake JDBC driver, snowflake-jdbc-3.2.2.jar in this example.

4. In the Driver Class field, enter the Snowflake JDBC driver class, `com.snowflake.client.jdbc.SnowflakeDriver` in this example.

5. In the Username and Password fields, enter the authentication information accordingly.

Creating a Snowflake table and a Snowflake sequence

Procedure

1. Double-click the first `tJDBCRow` component to open its Basic settings view.
2. Select the **Use an existing connection** check box and from the **Component List** drop-down list displayed, select the connection component to reuse the connection created by it, `tJDBCConnection_1` in this example.

3. In the **Query** field, enter the SQL command used to create a new table.

   In this example, the SQL command is `CREATE OR REPLACE TABLE employee (id INTEGER, name VARCHAR(50), role VARCHAR(50), salary DOUBLE, PRIMARY KEY(id))`, which creates a new table `employee` with four columns, `id` of INTEGER type as the primary key, `name` and `role` of VARCHAR type, and `salary` of DOUBLE type. This table will be used to store the employee data.

4. Double-click the second `tJDBCRow` component to open its **Basic settings** view.

5. Select the **Use an existing connection** check box and from the **Component List** drop-down list displayed, select the connection component to reuse the connection created by it, `tJDBCConnection_1` in this example.

6. In the **Query** field, enter the SQL command used to create a Snowflake sequence.

   In this example, the SQL command is `create or replace sequence employee_sequence`, which creates a new sequence `employee_sequence`. This sequence will be used by the `tJDBCSCDELT` component to generate the surrogate key for SCD Type 2 method.

**Inserting data into the new Snowflake table**

**Procedure**

1. Double-click the first `tFixedFlowInput` component to open its **Basic settings** view.

2. Click the `[...]` button next to **Edit schema** and in the pop-up dialog box, define the schema by adding four columns: `id` of Integer type as the primary key, `name` and `role` of String type, and `salary` of Double type.

3. Click **OK** to save the schema changes. In the pop-up dialog box, click **Yes** to propagate the schema to the next component.

4. Select **Use Inline Content** in the **Mode** area. Then in the **Content** field displayed, enter the following employee data to be inserted.

   111;Mark Smith;tester;15000.00
   222;Thomas Johnson;developer;18000.00
   333;Teddy Brown;tester;16000.00
5. Double-click the first **tJDBCOutput** component to open its **Basic settings** view.

6. Select the **Use an existing connection** check box and from the **Component List** drop-down list displayed, select the connection component to reuse the connection created by it, **tJDBCConnection_1** in this example.

7. In the **Table** field, enter the name of the table into which the employee data will be written, **employee** in this example.

8. In the **Action on data** drop-down list, select **Insert** to insert the employee data transferred from the first **tFixedFlowInput** component.

**Tracking data insertion changes and writing the changes into a SCD dimension table**

**Procedure**

1. Double-click the first **tJDBCSCDELT** component to open its **Basic settings** view.

2. Select the **Use an existing connection** check box and from the **Component List** drop-down list displayed, select the connection component to reuse the connection created by it, **tJDBCConnection_1** in this example.
3. In the **Source table** field, enter the name of the table whose data changes will be captured, `employee` in this example.

4. In the **Table** field, enter the name of the SCD dimension table that will store both the current and historical employee data, `employee_scd` in this example.

5. Select **Drop table if exists and create** from the **Action on table** drop-down list to create the SCD dimension table.

6. Click the [...] button next to **Edit schema** and in the pop-up dialog box, define the schema by adding nine columns: `sk` and `id` of Integer type as the primary key, `name` and `role` of String type, `salary` of Double type, `start_date` and `end_date` of Date type with the Date Pattern `yyyy-MM-dd`, and `active_status` and `version` of Integer type. When done, click **OK** to save the changes and close the dialog box.

7. From the **Surrogate key** drop-down list, select the name of the column that will be used as the primary key of the SCD dimension table, `sk` in this example.

8. Select **DB sequence** from the **Creation** drop-down list and in the **Sequence** field displayed, enter the name of the Snowflake sequence used to generate the surrogate key for the SCD Type 2 method.

9. Click the [...] button below the **Source keys** table to add a new line, and click the **Name** cell and select the key column of the source table from the drop-down list, `id` in this example.

10. Select the **Use SCD type 1 fields** check box, click the [...] button below the **SCD type 1 fields** table twice to add two lines. Then click each cell and from the drop-down list, select the column on which the SCD Type 1 method will be performed. In this example, they are `name` and `role`.

11. Select the **Use SCD type 2 fields** check box, click the [...] button below the **SCD type 2 fields** table to add a line. Then click the cell and select the column on which the SCD Type 2 method will be performed. In this example, it is `salary`.

12. From the **Start date** and **End date** drop-down lists, select the columns used to hold the start date and end date values for the SCD Type 2 method respectively, `start_date` and `end_date` in this example.
13. Select the **Log active status** check box and from the **Active field** drop-down list displayed, select the column used to hold the active status value for the SCD Type 2 method, which helps identify the active records, `active_status` in this example.

14. Select the **Log versions** check box and from the **Version field** drop-down list, select the column used to hold the version number of the records for the SCD Type 2 method, `version` in this example.

15. Select **Mapping Snowflake** from the **Mapping** drop-down list to use the Snowflake metadata mapping file.

**Retrieving the data insertion updates from the SCD dimension table**

**Procedure**

1. Double-click the first **tJDBCInput** component to open its **Basic settings** view.

2. Select the **Use an existing connection** check box and from the **Component List** drop-down list displayed, select the connection component to reuse the connection created by it, **tJDBCConnection_1** in this example.

3. Click the [...] button next to **Edit schema** and in the pop-up dialog box, define the schema by adding nine columns: `sk` and `id` of Integer type as the primary key, `name` and `role` of String type, `salary` of Double type, `start_date` and `end_date` of Date type with the Date Pattern `yyyy-MM-dd`, and `active_status` and `version` of Integer type. When done, click **OK** to save the changes and close the dialog box.

   The schema of the first **tJDBCInput** component is the same as the schema of the **tJDBCSCDELT** component, you can just copy and paste it.

4. In the **Query** field, enter the SQL command used to retrieve data from the SCD dimension table, `select * from employee_scd` in this example.

5. Double-click the first **tLogRow** component and in the **Mode** area on its **Basic settings** view, select **Table** to display the retrieved data in a table.

**Updating data in the Snowflake table**

**Procedure**

1. Double-click the second **tFixedFlowInput** component to open its **Basic settings** view.

2. Click the [...] button next to **Edit schema** and in the pop-up dialog box, define the schema by adding four columns: `id` of Integer type as the primary key, `name` and `role` of String type, and `salary` of Double type.

   This schema is the same as the schema of the first **tFixedFlowInput** component, you can just copy and paste it.

3. Click **OK** to save the schema changes. In the pop-up dialog box, click **Yes** to propagate the schema to the next component.

4. Select **Use Inline Content** in the **Mode** area. Then in the **Content** field displayed, enter the following employee data to update the existing data.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Role</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>Mark Smith</td>
<td>tester</td>
<td>15000.00</td>
</tr>
<tr>
<td>222</td>
<td>Thomas Johnson</td>
<td>tester</td>
<td>18000.00</td>
</tr>
<tr>
<td>333</td>
<td>Teddy Brown</td>
<td>writer</td>
<td>17000.00</td>
</tr>
</tbody>
</table>
5. Double-click the second `tJDBCOutput` component to open its Basic settings view.

6. Select the Use an existing connection check box and from the Component List drop-down list displayed, select the connection component to reuse the connection created by it, `tJDBCConnection_1` in this example.

7. In the **Table** field, enter the name of the table, in which the data will be updated, `employee` in this example.

8. Select **Insert or update** from the Action on data drop-down list.

**Tracking data update changes and writing the changes into the SCD dimension table**

**Procedure**

1. Double-click the second `tJDBCSCDELT` component to open its Basic settings view.

2. Repeat 2 on page 1896 through 15 on page 1898 in the procedure Tracking data insertion changes and writing the changes into a SCD dimension table on page 1896 to configure the second `tJDBCSCDELT` component.

**Retrieve the data update changes from the SCD dimension table**

**Procedure**

1. Double-click the second `tJDBCInput` component to open its Basic settings view.

2. Repeat 2 on page 1898 through 4 on page 1898 in the procedure Retrieving the data insertion updates from the SCD dimension table on page 1898 to configure the second `tJDBCInput` component.

3. Double-click the second `tLogRow` component and in the Mode area on its Basic settings view, select **Table** to display the retrieved data in a table.

**Executing the Job to track data changes in a Snowflake table using tJDBCSCDELT**

**Procedure**

1. Press `Ctrl + S` to save the Job.

2. Press `F6` to execute the Job.
As shown above, the old role developer for Thomas Johnson is overwritten directly by the new role tester because SCD Type 1 is performed on the role column, and a new record with the surrogate key value set to 4 is created for Teddy Brown’s salary update from 16000.00 to 17000.00 because SCD Type 2 is performed on the salary column.
tJDBCSP

Centralizes multiple or complex queries in a database in order to call them easily.

**tJDBCSP Standard properties**

These properties are used to configure tJDBCSP running in the Standard Job framework.

The Standard tJDBCSP component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in. This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
<td></td>
</tr>
<tr>
<td><strong>Connection Component</strong></td>
<td>Select the component that opens the database connection to be reused by this component.</td>
</tr>
<tr>
<td><strong>JDBC URL</strong></td>
<td>The JDBC URL of the database to be used. For example, the JDBC URL for the Amazon Redshift database is <code>jdbc:redshift://endpoint:port/database</code>.</td>
</tr>
<tr>
<td><strong>Drivers</strong></td>
<td>Complete this table to load the driver JARs needed. To do this, click the [+] button under the table to add as many rows as needed, each row for a driver JAR, then select the cell and click the [...] button at the right side of the cell to open the Module wizard from which you can select the driver JAR of your interest. For example, the driver jar <code>Redshift.JDBC41-1.1.13.1013.jar</code> for the Redshift database.</td>
</tr>
<tr>
<td><strong>Driver Class</strong></td>
<td>The class name for the specified driver between double quotation marks. For example, for the RedshiftJDBC41-1.1.13.1013.jar driver, the name to be entered is com.amazon.redshift.jdbc41.Driver.</td>
</tr>
<tr>
<td><strong>Use Id and Password</strong></td>
<td>The database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>SP Name</strong></td>
<td>The name of the stored procedure.</td>
</tr>
<tr>
<td><strong>Is function</strong></td>
<td>Select this check box if the stored procedure is a function and a value will be returned from the stored procedure. With this check box selected, you need to select the column in which the returned value will be saved from the Return result in drop-down list.</td>
</tr>
</tbody>
</table>
| **Set SP Parameters** | Specify the parameters for the stored procedure by clicking the [+] button to add as many rows as needed, each row for a parameter, and setting the following value for each parameter.  
  - **Schema Column**: The column that holds the parameter value.  
  - **Parameter Type**: The type of the parameter.  
    - **IN**: The input parameter.  
    - **OUT**: The output parameter/return value.  
    - **IN OUT**: The combination of input and output parameters. It means the parameter can be passed to the stored procedure, modified by the procedure, and returned with a new value.  
    - **RECORDSET**: The input parameter will be returned as a set of values, rather than a single value. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
  - **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.  
  - **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.  
  Click Edit schema to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.  
  - **View schema**: choose this option to view the schema only. |
• **Change to built-in property**: choose this option to change the schema to Built-in for local changes.

• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Specify a data source alias**

Select this check box and in the Data source alias field displayed, specify the alias of a data source created on Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

If you use the component’s own DB configuration, your data source connection will be closed at the end of the component. To prevent this from happening, use a shared DB connection with the data source alias specified.

This property is not available when other connection component is selected from the Connection Component drop-down list.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**ERROR_MESSAGE**

The error message generated by the component when an error occurs. This is an After variable and it returns a string.

### Usage

**Usage rule**

This component is used as intermediary component. It can be used as start component but only input parameters are thus allowed.

**Dynamic settings**

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example,
when your Job has to be deployed and executed independent of Talend Studio.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

| Limitation | The Stored Procedures syntax should match the Database syntax. |

**Related scenario**

For related scenarios, see:

- Executing a stored procedure using tMDMSP on page 2190.
- Scenario: Checking number format using a stored procedure on page 2809

Check as well Scenario: Inserting data in mother/daughter tables on page 2476 if you want to analyze a set of records from a database table or DB query and return single records.
tJDBCTableList

Lists the names of a given set of JDBC tables using a select statement based on a Where clause. tJDBCTableList iterates on a set of table names through a defined JDBC connection.

**tJDBCTableList Standard properties**

These properties are used to configure tJDBCTableList running in the Standard Job framework. The Standard tJDBCTableList component belongs to the Databases family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Database type</th>
<th>Select a database from the list, such as Mysql, Oracle, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tJDBCConnection component or a database-specific connection component if the relevant database is selected from the Database type list.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Use filter</th>
<th>Select this check box to filter the tables to be iterated on.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular expression for tables name</td>
<td>Enter the regular expression to identify table names. Available when Use filter is selected.</td>
</tr>
<tr>
<td>Filter criteria</td>
<td>Select the criteria for filtering table names. Include: iterates on the table names identified by the regular expression only. Exclude: iterates on the table names other than those identified by the regular expression. Available when Use filter is selected.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect the log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>CURRENT_TABLE: the name of the table currently iterated upon. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_TABLE: the number of tables iterated upon so far. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions</td>
</tr>
</tbody>
</table>
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage rule</strong></td>
</tr>
</tbody>
</table>

**Related scenario**

For **tJDBCTableList** related scenario, see *Scenario: Iterating on a DB table and listing its column names* on page 2470.
tJIRAInput

Retrieves the issue information based on a JQL query or retrieve the project information based on a specified project ID from JIRA.

**tJIRAInput Standard properties**

These properties are used to configure tJIRAInput running in the Standard Job framework.

The Standard tJIRAInput component belongs to the Business family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Host URL</th>
<th>Specify the URL used to access JIRA.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User Id and Password</strong></td>
<td>Specify the user authentication data used to access JIRA. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>JIRA resource</strong></td>
<td>Select the JIRA resource from the drop-down list, either ISSUE or PROJECT.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. Note that the schema of this component is read-only with a single column json of String type. It stores the issue or project information in JSON format. You can click Edit schema to view the schema.</td>
</tr>
<tr>
<td><strong>JQL</strong></td>
<td>Specify the JQL (JIRA Query Language) query based on which you will search for issues in JIRA. For more information about the JQL, see <a href="https://confluence.atlassian.com/jirasoftware/072/advanced-searching-829057400.html">https://confluence.atlassian.com/jirasoftware/072/advanced-searching-829057400.html</a>. This field is available only when ISSUE is selected from the Jira resource drop-down list.</td>
</tr>
<tr>
<td><strong>Project ID</strong></td>
<td>Enter the ID of the project whose information you want to retrieve from JIRA. This field is available only when PROJECT is selected from the Jira resource drop-down list.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| Batch size | Specify the number of issues to be processed in each batch. |
tJIRAInput

This field is available only when ISSUE is selected from the Jira resource drop-down list.

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

- **Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is usually used as a start component of a Job or Subjob and it always needs an output link.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>This component doesn’t support retrieving the comments of issues from JIRA.</td>
</tr>
</tbody>
</table>

### Retrieving the project information from JIRA application

Here is an example of using Talend components to retrieve the information of a project from JIRA application, extract the information needed based on the JSONPath query, and finally write the extracted information into a JSON file.

#### Creating a Job for retrieving the project information from JIRA application

![Diagram](Retrieve the project information)

**Before you begin**

A project with the key DOC has been created in JIRA application.
**Procedure**

1. Create a new Job and add a tJIRAInput component, a tExtractJSONFields component, and a tFileOutputJSON component to the Job.
2. Link the tJIRAInput component to the tExtractJSONFields component using a Row > Main connection.
3. Link the tExtractJSONFields component to the tFileOutputJSON component using a Row > Main connection.

**Configuring the Job for retrieving the project information from JIRA application**

**Procedure**

1. Double-click the tJIRAInput component to open its Basic settings view.

   - In the Host URL, User Id, and Password fields, specify the user authentication information required to access the JIRA application.
2. Select Project from the JIRA resource drop-down list, in the Project ID field displayed, enter the key of the project, whose information will be retrieved. In this example, it is DOC.
3. Double-click the tExtractJSONFields component to open its Basic settings view.

   - Mapping:
     - Column | Json query |
       - id     | "$id"      |
       - key    | "$key"     |
       - name   | "$name"    |
       - description | "$description" |
       - assigneeType | "$assigneeType" |
       - projectTypeKey | "$projectTypeKey" |

   - Die on error
5. Click the [...] button next to Edit schema, and in the schema dialog box displayed, define the
schema by adding six columns of String type, id, key, name, description, assigneeType, and
ProjectTypeKey, which will hold the corresponding information of the specified project.

When done, click OK to save the changes and close the dialog box.

6. Select the JSON field to be extracted from the JSON field drop-down list. In this example, it is json,
which is passed from the tJIRAInput component.

7. In the Loop Jsonpath query field, specify the path to the node on which the loop is based. In this
example, it is $, which is the root node of the JSON field.

8. In the Mapping table, the Column cells are automatically filled with the schema columns, and you
need to specify the corresponding JSON node that holds the desired data for each schema column.
In this example, $.id for the id column, $.key for the key column, $.name for the name column,
$.description for the description column, $.assigneeType for the assigneeType column, and
$.projectTypeKey for the projectTypeKey column.

9. Double-click the tFileOutputJSON component to open its Basic settings view.

10.In the File Name field, specify the path to the file into which the extracted project information will
be written, D:/JiraComponents/project_info.json in this example. And in the Name of data
block field, enter the name of the data block, project in this example.

Executing the Job to retrieve the project information from JIRA application

Procedure

1. Press Ctrl + S to save the Job.
2. Press F6 to execute the Job.
3. Open the JSON file project_info.json generated.

Note that all extracted project information is displayed in one line in the generated JSON file, and
you can use some editor tool to format and indent the data into an organized manner for better
display of the result.

As shown above, the desired project information has been successfully extracted and written into
the JSON file.
tJIRAOutput

Inserts, updates, or deletes the issue or project information in JIRA.

**tJIRAOutput Standard properties**

These properties are used to configure tJIRAOutput running in the Standard Job framework.

The Standard tJIRAOutput component belongs to the Business family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host URL</strong></td>
<td>Specify the URL used to access JIRA.</td>
</tr>
<tr>
<td><strong>User Id and Password</strong></td>
<td>Specify the user authentication data used to access JIRA. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>JIRA resource</strong></td>
<td>Select the JIRA resource from the drop-down list, either ISSUE or PROJECT.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. Note that the schema of this component is read-only with a single column json of String type. It stores the issue or project information in JSON format. You can click Edit schema to view the schema.</td>
</tr>
<tr>
<td><strong>Output Action</strong></td>
<td>Select an operation to be performed from the drop-down list.</td>
</tr>
<tr>
<td></td>
<td>- <strong>INSERT</strong>: insert new issue or project information into JIRA.</td>
</tr>
<tr>
<td></td>
<td>- <strong>UPDATE</strong>: modify existing issue or project information in JIRA.</td>
</tr>
<tr>
<td></td>
<td>- <strong>DELETE</strong>: remove issue or project data in JIRA corresponding to the input flow.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Delete subtasks</strong></td>
<td>Select this check box to delete the subtask(s). This check box is available only when ISSUE is selected from the JIRA resource drop-down list and DELETE is selected from the Output Action drop-down list.</td>
</tr>
</tbody>
</table>
### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_SUCCESS: the number of rows successfully processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_REJECT: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

| Usage rule | This component is usually used as an end component of a Job or Subjob and it always needs an input link. |

### Creating an issue in JIRA application

Here is an example of using Talend components to create a new issue in JIRA application, then retrieve the information of the new issue from JIRA application, finally extract the information needed based on the JSONPath query and write the extracted information into a JSON file.

### Setting up a Job for creating an issue in JIRA application
Before you begin

A project with the key DOC has been created in JIRA application and only one issue with the key DOC-1 has already been created under this project.

Procedure

1. Create a new Job and add a tFileInputDelimited component, a tJIRAOutput component, a tJIRAInput component, a tExtractJSONFields component, and a tFileOutputJSON component to the Job.
2. Double-click the tJIRAOutput component to open its Basic settings view.
3. Select Issue from the JIRA resource drop-down list and Insert from the Output Action drop-down list to insert an issue.
4. Link the tFileInputDelimited component to the tJIRAOutput component using a Row > Main connection. In the pop-up dialog box, click Yes to let the tFileInputDelimited component get the schema of the tJIRAOutput component.
5. Do the same to link the tJIRAInput component to the tExtractJSONFields component and the tExtractJSONFields component to the tFileOutputJSON component.
6. Link the tFileInputDelimited component to the tJIRAInput component using a Trigger > On Subjob Ok connection.

Creating a new issue in JIRA application

Procedure

1. Double-click the tFileInputDelimited component to open its Basic settings view.
2. In the File name/Stream field, specify the path to the JSON file used to create the issue. In this example, a simple JSON file D:/JiraComponents/issue_create.json will be used to insert a new issue under the project with the key DOC and the data in this file reads as follows:

```json
{"fields": {
    "issuetype": {
        "id": "10001",
        "name": "Task"
    },
    "project": {
        "key": "DOC",
        "name": "Documentation"
    }
}}
```
Note that the `tJIRAOutput` component now only supports creating project/issue in JIRA application with the single line JSON file, so you need to join all lines in the input JSON file into one line before executing the Job.

For more information about the types of issues, see the JIRA official documentation `Issue types`.

For more information about how to find the id for issue types, see the JIRA official documentation `Finding the Id for Issue Types`.

3. Double-click the `tJIRAOutput` component to open its `Basic settings` view.
4. In the `Host URL`, `User Id`, and `Password` fields, specify the user authentication information required to access the JIRA application.

**Retrieving the new issue information from JIRA application**

**Procedure**

1. Double-click the `tJIRAInput` component to open its `Basic settings` view.

2. Select `Issue` from the `JIRA resource` drop-down list.
3. In the `JQL` field, enter the JQL query used to retrieve issues in JIRA application.

   In this example, the query is `project = DOC ORDER BY created DESC`. All issues under the project `DOC` will be retrieved and they will be sorted by the creation date in descending order.
4. Double-click the `tExtracJSONFields` component to open its `Basic settings` view.
5. Click the [...] button next to Edit schema, and in the schema dialog box displayed, define the schema by adding five columns of String type, id, key, project, description, and summary, which will hold the corresponding information of each issue retrieved. When done, click OK to save the changes and close the dialog box.

6. Select the JSON field to be extracted from the JSON field drop-down list. In this example, it is json, which is passed from the tJIRAInput component.

7. In the Loop Jsonpath query field, specify the path to the node on which the loop is based. In this example, it is $, which is the root node of the JSON field.

8. In the Mapping table, the Column cells are automatically filled with the schema columns, and you need to specify the corresponding JSON node that holds the desired data for each schema column. In this example, $.id for the id column, $.key for the key column, $.fields.project.key for the project column, $.fields.description for the description column, and $.fields.summary for the summary column.

9. Double-click the tFileOutputJSON component to open its Basic settings view.

10. In the File Name field, specify the path to the file into which the extracted issue information will be written, D:/JiraComponents/issue_info.json in this example. And in the Name of data block field, enter the name of the data block, issue in this example.

**Executing the Job to create an issue in JIRA application**

**Procedure**

1. Press Ctrl + S to save the Job.
2. Press F6 to execute the Job.
3. Open the JSON file issue_info.json generated.

   Note that all extracted issue information is displayed in one line in the generated JSON file, and you can use some editor tool to format and indent the data into an organized manner for better display of the result.
As shown above, a new issue with the key DOC-2 has been created, and the desired information for each issue in the project DOC has been successfully extracted and written into the JSON file.

Updating an issue in JIRA application

Here is an example of using Talend components to update an existing issue in JIRA application, then retrieve the information of the updated issue from JIRA application, finally extract the information needed based on the JSONPath query and write the extracted information into a JSON file.

Creating a Job for updating an issue in JIRA application

Before you begin

A project with the key DOC has been created in JIRA application and an issue with the key DOC-2 as shown below has already been created under this project.
Procedure

1. Create a new Job and add a **tFileInputDelimited** component, a **tJIRAOutput** component, a **tJIRAInput** component, a **tExtractJSONFields** component, and a **tFileOutputJSON** component to the Job.

2. Double-click the **tJIRAOutput** component to open its **Basic settings** view.

3. Select **Issue** from the **JIRA resource** drop-down list and **Update** from the **Output Action** drop-down list to update an issue.

4. Link the **tFileInputDelimited** component to the **tJIRAOutput** component using a **Row > Main** connection. In the pop-up dialog box, click **Yes** to let the **tFileInputDelimited** component get the schema of the **tJIRAOutput** component.

5. Do the same to link the **tJIRAInput** component to the **tExtractJSONFields** component and the **tExtractJSONFields** component to the **tFileOutputJSON** component.

6. Link the **tFileInputDelimited** component to the **tJIRAInput** component using a **Trigger > On Subjob Ok** connection.

Updating an issue in JIRA application

**Procedure**

1. Double-click the **tFileInputDelimited** component to open its **Basic settings** view.

2. In the **File name/Stream** field, specify the path to the JSON file used to update the issue.

   In this example, a simple JSON file `D:/JiraComponents/issue_update.json` will be used to update an existing issue with the key `DOC-2` that is under the `DOC` project, and the data in this file reads as follows:

   ```json
   DOC-2;
   {"fields": {
       "description": "Update an issue under the DOC project",
       "summary": "Add the documentation for new components tJIRAInput and tJIRAOutput"
   }}
   ```

   Note that the **tJIRAOutput** component now only supports updating project/issue in JIRA application with the single line JSON file, so you need to join all lines in the input JSON file into one line before executing the Job.

3. Double-click the **tJIRAOutput** component to open its **Basic settings** view.

4. In the **Host URL**, **User Id**, and **Password** fields, specify the user authentication information required to access the JIRA application.

**Retrieving the updated issue information from JIRA application**
Procedure

1. Double-click the **tJIRAInput** component to open its **Basic settings** view.

![tJIRAInput 1](image)

- **Host URL**: [redacted]
- **User Id**: [redacted]
- **Password**: [redacted]
- **JIRA resource**: Issue
- **Schema**: Built-In
- **JQL**: “project = DOC AND issuekey = DOC-2”

2. Select **Issue** from the **JIRA resource** drop-down list.

3. In the **JQL** field, enter the JQL query used to retrieve the issue in JIRA application.
   
   In this example, the query is `project = DOC AND issuekey = DOC-2`. The issue with the key **DOC-2** under the project **DOC** will be retrieved.

4. Double-click the **tExtractJSONFields** component to open its **Basic settings** view.

![tExtractJSONFields 1](image)

5. Click the [...] button next to **Edit schema**, and in the schema dialog box displayed, define the schema by adding five columns of **String** type, **id**, **key**, **project**, **description**, and **summary**, which will hold the corresponding information of each issue retrieved.

   When done, click **OK** to save the changes and close the dialog box.

6. Select the JSON field to be extracted from the **JSON field** drop-down list. In this example, it is **json**, which is passed from the **tJIRAInput** component.

7. In the **Loop Jsonpath query** field, specify the path to the node on which the loop is based. In this example, it is **$**, which is the root node of the JSON field.

8. In the **Mapping** table, the **Column** cells are automatically filled with the schema columns, and you need to specify the corresponding JSON node that holds the desired data for each schema column. In this example, **$.id** for the **id** column, **$.key** for the **key** column, **$.fields.project.key** for the

---

1918
9. Double-click the **tFileOutputJSON** component to open its **Basic settings** view.

10. In the **File Name** field, specify the path to the file into which the extracted issue information will be written, `D:/JiraComponents/issue_info_update.json` in this example. And in the **Name of data block** field, enter the name of the data block, `issue` in this example.

### Executing the Job to update an issue in JIRA application

#### Procedure

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to execute the Job.
3. Open the JSON file `issue_info_update.json` generated.

   Note that all extracted issue information is displayed in one line in the generated JSON file, and you can use some editor tool to format and indent the data into an organized manner for better display of the result.

   ```json
   { "issue": [{
   "summary": "Add the documentation for new components tJIRAInput and tJIRAOoutput",
   "project": "DOC",
   "description": "Update an issue under the DOC project",
   "id": "10604",
   "key": "DOC-2"
   }]
   }
   ```

As shown above, the issue with the key `DOC-2` has been updated, and the desired information for the issue has been successfully extracted and written into the JSON file.
tJMSInput

Creates an interface between a Java application and a Message-Oriented middleware system.

Using a JMS server, tJMSInput makes it possible to have loosely coupled, reliable, and asynchronous communication between different components in a distributed application.

**tJMSInput Standard properties**

These properties are used to configure tJMSInput running in the Standard Job framework.

The Standard tJMSInput component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Module List</th>
<th>Select the library to be used from the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Provider</td>
<td>Type in the context URL, for example com.tibco.tibjms.naming.TibjmsInitialContextFactory. However, be careful, the syntax can vary according to the JMS server used.</td>
</tr>
<tr>
<td>Server URL</td>
<td>Type in the server URL, respecting the syntax, for example tibjmsnaming://localhost:7222.</td>
</tr>
<tr>
<td>Connection Factory JDNI Name</td>
<td>Type in the JDNI name.</td>
</tr>
<tr>
<td>Use Specified User Identity</td>
<td>If you have to log in, select the check box and type in your login and password. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Enable Durable Subscription</td>
<td>Select this check box to enable the durable subscription.</td>
</tr>
<tr>
<td>ClientID</td>
<td>Enter the client ID for the durable subscription. This field is available only when the Enable Durable Subscription check box is selected.</td>
</tr>
<tr>
<td>Subscriber Name</td>
<td>Enter the subscriber name for the durable subscription. This field is available only when the Enable Durable Subscription check box is selected.</td>
</tr>
<tr>
<td>Use JNDI Name Lookup Destination</td>
<td>Select this check box to look up a destination with the JNDI (Java Naming and Directory Interface) name.</td>
</tr>
<tr>
<td>Message Type</td>
<td>Select the message type, either: Topic or Queue.</td>
</tr>
<tr>
<td><strong>Message From</strong></td>
<td>Type in the message source, exactly as expected by the server; this must include the type and name of the source. e.g.: queue/A or topic/testtopic. Note that the field is case-sensitive.</td>
</tr>
<tr>
<td><strong>Timeout for Next Message (in sec)</strong></td>
<td>Type in the number of seconds before passing to the next message.</td>
</tr>
<tr>
<td><strong>Maximum Messages</strong></td>
<td>Type in the maximum number of messages to be processed.</td>
</tr>
<tr>
<td><strong>Message Selector Expression</strong></td>
<td>Set your filter.</td>
</tr>
<tr>
<td><strong>Processing Mode</strong></td>
<td>Select the processing mode for the messages. Raw Message or Message Content.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. The schema of this component is read-only. You can click Edit schema to view the schema.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **Properties** | Click the plus button underneath the table to add lines that contain username and password required for user authentication. |
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

| **Global Variables** | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  

NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  

For further information about variables, see Talend Studio User Guide. |

1921
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is generally used as an input component. It must be linked to an output component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Make sure the JMS server is launched. Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

Related scenarios

For related scenarios, see Scenario: Enqueuing/dequeuing a message on the ActiveMQ server on page 1925.
tJMSOutput

creates an interface between a Java application and a Message-Oriented middle ware system.

Using a JMS server, tJMSOutput makes it possible to have loosely coupled, reliable, and asynchronous communication between different components in a distributed application.

**tJMSOutput Standard properties**

These properties are used to configure tJMSOutput running in the Standard Job framework.
The Standard tJMSOutput component belongs to the Internet family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Module List</th>
<th>Select the library to be used from the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Provider</td>
<td>Type in the context URL, for example <code>com.tibco.tibms.naming.TibjmsInitialContextFactory</code>. However, be careful, the syntax can vary according to the JMS server used.</td>
</tr>
<tr>
<td>Server URL</td>
<td>Type in the server URL, respecting the syntax, for example <code>tibjmsnaming://localhost:7222</code>.</td>
</tr>
<tr>
<td>Connection Factory JDNI Name</td>
<td>Type in the JDNI name.</td>
</tr>
<tr>
<td>Use Specified User Identity</td>
<td>If you have to log in, select the check box and type in your login and password. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Message Type</td>
<td>Select the message type, either: Topic or Queue.</td>
</tr>
<tr>
<td>To</td>
<td>Type in the message target, as expected by the server.</td>
</tr>
<tr>
<td>Processing Mode</td>
<td>Select the processing mode for the messages. Raw Message or Message Content</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The tJMSOutput schema is read-only. It is made of one column: Message</td>
</tr>
</tbody>
</table>

**Advanced settings**

| Delivery Mode                                                             | Select a delivery mode from this list to ensure the quality of data delivery: |
**tMSOutput**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Not Persistent: This mode allows data loss during the data exchange. Persistent: This mode ensures the integrity of message delivery.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Properties</strong> Click the plus button underneath the table to add lines that contains username and password required for user authentication.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</th>
</tr>
</thead>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is generally used as an output component. It must be linked to an input component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Make sure the JMS server is launched. Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>
Scenario: Enqueuing/dequeuing a message on the ActiveMQ server

In this scenario, JMSOutput sends a message to a queue on the ActiveMQ server, which is then retrieved by JMSInput. This message is finally displayed on the console via tLogRow.

Linking the components

Procedure

1. Drop tFixedFlowInput, JMSOutput, JMSInput, and tLogRow onto the workspace.
2. Link tFixedFlowInput to JMSOutput using a Row > Main connection.
3. Link tFixedFlowInput to JMSInput using the OnSubjobOk trigger.
4. Link JMSInput to tLogRow using a Row > Main connection.

Configuring the components

Procedure

1. Double-click tFixedFlowInput to open its Basic settings view.

<table>
<thead>
<tr>
<th>Basic settings</th>
<th>Schema</th>
<th>Built-In</th>
<th>Edit schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced settings</td>
<td>Number of rows</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Dynamic settings</td>
<td>Mode</td>
<td>Use Inline Table</td>
<td></td>
</tr>
<tr>
<td>View</td>
<td>Use Inline Content (delimited file)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>Row Separator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validation Rules</td>
<td>Content</td>
<td>message transferred</td>
<td></td>
</tr>
</tbody>
</table>

Select Use Inline Content (delimited file) in the Mode area.
In the **Content** field, enter the content of the message to be sent to the ActiveMQ server:

```
message transferred
```

2. Click the **Edit schema** button to open the schema editor.

3. Click the `[+]` button to add one column, namely *messageContent*, of the string type. Click **OK** to validate the setup and close the editor.

4. Now appears the pop-up box that asks for schema propagation.

```
Would you like to propagate changes?
```

Click **Yes** to propagate the schema to the subsequent component.

5. Double-click **JMSOutput** to open its **Basic settings** view.

6. In the **Module List** list, select the library to be used, namely the *activemq* jar in this case.
7. In the **Context Provider** field, enter the context URI, "org.apache.activemq.jndi.ActiveMQInitialContextFactory" in this case.

8. In the **Server URL** field, enter the ActiveMQ Server URI.

9. In the **Connection Factory JDNI Name** field, enter the JDNI name, "QueueConnectionFactory" in this case.

10. Select the **Use Specified User Identity** check box to show the **User Name** and **Password** fields, where you can enter the authentication data.

11. In the **Message type** list, select **Queue**.

12. In the **Processing Mode** list, select **Message Content**.

13. Perform the same setup in the **Basic settings** view of JMSInput.

---

**Executing the Job**

**Procedure**

1. Press **Ctrl + S** to save the Job.

2. Press **F6** to run the Job. Note that the ActiveMQ server has started at tcp://192.168.30.46:61616.

   ```
   [statistics] connecting to socket on port 3999
   [statistics] connected
   Ready to receive message
   Waiting...
   message transferred
   ```

   As shown above, the message is correctly transferred and displayed.

**Related scenarios**

For related scenarios, see Scenario 1: Asynchronous communication via a MOM server on page 2283 and Scenario 2: Transmitting XML files via a MOM server on page 2285.
tJoin

Performs inner or outer joins between the main data flow and the lookup flow.

tJoin joins two tables by doing an exact match on several columns. It compares columns from the main flow with reference columns from the lookup flow and outputs the main flow data and/or the rejected data.

tJoin Standard properties

These properties are used to configure tJoin running in the Standard Job framework.
The Standard tJoin component belongs to the Processing family.
The component in this framework is generally available.

Basic settings

| Schema and Edit schema | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:
|                       | • View schema: choose this option to view the schema only.
|                       | • Change to built-in property: choose this option to change the schema to Built-in for local changes.
|                       | • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
| Built-in: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide. | |
| Repository: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job flowcharts. Related topic: see Talend Studio User Guide. | |
| Include lookup columns in output | Select this check box to include the lookup columns you define in the output flow. |
| Input key attribute | Select the column(s) from the main flow that needs to be checked against the reference (lookup) key column. |
### Lookup key attribute

Select the lookup key columns that you will use as a reference against which to compare the columns from the input flow.

### Inner join (with reject output)

Select this check box to join the two tables first and gather the rejected data from the main flow.

### Advanced settings

**tStat Catcher Statistics**

Select this check box to collect log data at the component level.

### Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

**Usage rule**

This component is not startable and it requires two input components and one or more output components.

### Scenario 1: Doing an exact match on two columns and outputting the main and rejected data

This scenario describes a five-component Job aiming at carrying out an exact match between the **firstnameClient** column of an input file against the data of the reference input file, and the **lastnameClient** column against the data of the reference input file. The outputs of this exact match are written in two separate files: exact data are written in an Excel file, and inaccurate data are written in a delimited file.

In this scenario, we have already stored the input schemas of the input and reference files in the Repository. For more information about storing schema metadata in the **Repository** tree view, see *Talend Studio User Guide*. 
Dropping and linking the components

Procedure

1. In the Repository tree view, expand Metadata and the file node where you have stored the input schemas and drop the relevant file onto the design workspace. The [Components] dialog box appears.

2. Select tFileInputDelimited from the list and click OK to close the dialog box. The tFileInputDelimited component displays in the workspace. The input file used in this scenario is called ClientSample. It holds four columns including the two columns firstnameClient and lastnameClient we want to do the exact match on.

3. Do the same for the second input file you want to use as a reference, ClientSample_Update in this scenario.

4. Drop the following components from the Palette onto the design workspace: tJoin, tFileOutputExcel, and tFileOutputDelimited.

5. Connect the main and reference input files to tJoin using Main links. The link between the reference input file and tJoin appears as a lookup link on the design workspace.
6. Connect tJoin to tFileOutputExcel using the Main link and tJoin to tFileOutputDelimited using the Inner join reject link.

Configuring the components

Procedure

1. If needed, double-click the main and reference input files to display their Basic settings views. All their property fields are automatically filled in. If you do not define your input files in the Repository, fill in the details manually after selecting Built-in in the Property Type field.

2. Double click tJoin to display its Basic settings view and define its properties.

3. Click the Edit schema button to open a dialog box that displays the data structure of the input files, define the data you want to pass to the output components, three columns in this scenario, idClient, firstnameClient and lastnameClient, and then click OK to validate the schema and close the dialog box.

4. In the Key definition area of the Basic settings view of tJoin, click the plus button to add two columns to the list and then select the input columns and the output columns you want to do the exact matching on from the Input key attribute and Lookup key attribute lists respectively, firstnameClient and lastnameClient in this example.
5. Select the **Inner join (with reject output)** check box to define one of the outputs as inner join reject table.

6. Double click **tFileOutputExcel** to display its **Basic settings** view and define its properties.

   ![Basic settings](image)

   - **File Name**: D:\TOS-AI-r52701-v4.2.0NB\workspace\out.xls
   - **Sheet name**: Sheet1
   - **Include header**: checked
   - **Append existing file**: unchecked
   - **Is absolute Y pos.**: unchecked
   - **Font**: Default
   - **Define all columns auto size**: unchecked
   - **Define column auto size**: unchecked

   ![Basic settings](image)

7. Set the destination file name and the sheet name, and select the **Include header** check box.

8. Double click **tFileOutputDelimited** to display its **Basic settings** view and define its properties.

   ![Basic settings](image)

   - **File Name**: D:\TOS-AI-r52701-v4.2.0NB\workspace\out.csv
   - **Row Separator**: "\n"
   - **Field Separator**: ",,"
   - **Include Header**: checked
   - **Compress as zip file**: unchecked
   - **Synccolumns**: unchecked

9. Set the destination file name, and select the **Include header** check box.

### Saving and executing the Job

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Press **F6**, or click **Run** on the **Run** tab to execute the Job.
The output of the exact match on the *firstnameClient* and *lastnameClient* columns is written to the defined Excel file.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>idClient</td>
<td>firstnameClient</td>
<td>lastnameClient</td>
</tr>
<tr>
<td>2</td>
<td>28</td>
<td>Herbert</td>
<td>Eisenhower</td>
</tr>
<tr>
<td>3</td>
<td>56</td>
<td>Chester</td>
<td>Grant</td>
</tr>
</tbody>
</table>

The rejected data is written to the defined delimited file.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>idClient;firstnameClient;lastnameClient</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1; Dwight; Madison</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2; Franklin; Jackson</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3; Ronald; Buchanan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4; Bill; Cleveland</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5; William; Harrison</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6; William; Fillmore</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7; Harry; Adams</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8; Harry; McKinley</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9; Herbert; Reagan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>10; Lyndon; Jefferson</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>11; Bill; Jackson</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>12; John; Hayes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>13; Ulysses; Reagan</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
tKafkaCommit

Saves the current state of the tKafkaInput to which it is connected.

The tKafkaCommit component connects to a given tKafkaInput to perform a consumer offset commit. Note that the term ‘commit’ in this component means saving what messages are consumed by this tKafkaInput component at the moment of committing.

**tKafkaCommit Standard properties**

These properties are used to configure tKafkaCommit running in the Standard Job framework.
The Standard tKafkaCommit component belongs to the Internet family.
The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

| Commit offsets from | Select the tKafkaInput component from which the consumed messages are committed. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

**Usage**

| Usage rule | Although tKafkaCommit can be used as an end component in a Subjob flow or can be called independently as its own Subjob, it is typically used standalone to commit a batch of offsets in one go. If you need to commit offsets regularly, it is recommended to use the Auto-commit feature in tKafkaInput. |

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tKafkaConnection

Opens a reusable Kafka connection.

The tKafkaConnection component opens a connection to a given Kafka cluster so that the other Kafka component in Subjobs can reuse this connection.

**tKafkaConnection Standard properties**

These properties are used to configure tKafkaConnection running in the Standard Job framework.

The Standard tKafkaConnection component belongs to the Internet family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Version</th>
<th>Select the version of the Kafka cluster to be used.</th>
</tr>
</thead>
</table>
| Zookeeper quorum list | Enter the address of the Zookeeper service of the Kafka cluster to be used.  
The form of this address should be hostname:port. This information is the name and the port of the hosting node in this Kafka cluster.  
If you need to specify several addresses, separate them using a comma (,). |
| Broker list | Enter the addresses of the broker nodes of the Kafka cluster to be used.  
The form of this address should be hostname:port. This information is the name and the port of the hosting node in this Kafka cluster.  
If you need to specify several addresses, separate them using a comma (,). |
| Use SSL/TLS | Select this check box to enable the SSL or TLS encrypted connection.  
Then you need to use the tSetKeystore component in the same Job to specify the encryption information.  
For further information about tSetKeystore, see tSetKeystore on page 3532.  
This check box is available since Kafka 0.9.0.1. |
| Use Kerberos authentication | Use Kerberos authentication  
If the Kafka cluster to be used is secured with Kerberos, select this check box to display the related parameters to be defined:  
- **JAAS configuration path**: enter the path, or browse to the JAAS configuration file to be used by the Job to authenticate as a client to Kafka.  
  This JAAS file describes how the clients, the Kafka-related Jobs in terms of Talend, can connect to the... |
Kafka broker nodes, using either the kinit mode or the keytab mode. It must be stored in the machine where these Jobs are executed.

_Talend_ , Kerberos or Kafka does not provide this JAAS file. You need to create it by following the explanation in Configuring Kafka client depending on the security strategy of your organization.

- **Kafka brokers principal name**: enter the primary part of the Kerberos principal you defined for the brokers when you were creating the broker cluster. For example, in this principal `kafka/kafka1.hostname.com@EXAMPLE.COM`, the primary part to be used to fill in this field is `kafka`.

- **Set kinit command path**: Kerberos uses a default path to its kinit executable. If you have changed this path, select this check box and enter the custom access path.

  If you leave this check box clear, the default path is used.

- **Set Kerberos configuration path**: Kerberos uses a default path to its configuration file, the `krb5.conf` file (or `krb5.ini` in Windows) for Kerberos 5 for example. If you have changed this path, select this check box and enter the custom access path to the Kerberos configuration file.

  If you leave this check box clear, a given strategy is applied by Kerberos to attempt to find the configuration information it requires. For details about this strategy, see the Locating the `krb5.conf` Configuration File section in Kerberos requirements.

For further information about how a Kafka cluster is secured with Kerberos, see Authenticating using SASL.

This check box is available since Kafka 0.9.0.1.

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used standalone to create the Kafka connection that the other Kafka components can reuse.</th>
</tr>
</thead>
</table>

### Related scenarios

No scenario is available for the Standard version of this component yet.
tKafkaCreateTopic

Creates a Kafka topic that the other Kafka components can use.

This component allows you to visually issue the command to create a topic with various topic-level properties.

**tKafkaCreateTopic Standard properties**

These properties are used to configure tKafkaCreateTopic running in the Standard Job framework.

The Standard tKafkaCreateTopic component belongs to the Internet family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version</strong></td>
<td>Select the version of the Kafka cluster to be used.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Zookeeper quorum list</strong></td>
<td>Enter the address of the Zookeeper service of the Kafka cluster to be used.</td>
</tr>
<tr>
<td></td>
<td>The form of this address should be <em>hostname:port</em>. This information is the name and the port of the hosting node in this Kafka cluster.</td>
</tr>
<tr>
<td></td>
<td>If you need to specify several addresses, separate them using a comma (,).</td>
</tr>
<tr>
<td><strong>Action on topic</strong></td>
<td>Select how a topic is created.</td>
</tr>
<tr>
<td></td>
<td>• Create topic: it creates a topic. If this topic exists already, the entire Job is stopped.</td>
</tr>
<tr>
<td></td>
<td>• Create topic if not exists: it creates a topic when this topic does not exist. If this topic exists, the Job skip the creation to move to the next step.</td>
</tr>
<tr>
<td><strong>Topic name</strong></td>
<td>Enter the name of the topic to be created.</td>
</tr>
<tr>
<td><strong>Replication factor</strong></td>
<td>Enter the number of the replicas to be created for the logs (the message sets) of the topic partitions.</td>
</tr>
<tr>
<td><strong>Number of partitions</strong></td>
<td>Enter the number of partitions to be created for the topic.</td>
</tr>
<tr>
<td><strong>Set topic retention time (ms)</strong></td>
<td>Select this check box to define the maximum time in milliseconds during which the contents of the topic are retained. At the end of the time, the contents are removed or compacted depending on the policy you define in the Topic properties table in the Advanced settings view.</td>
</tr>
</tbody>
</table>
Advanced settings

<table>
<thead>
<tr>
<th>Topic properties</th>
<th>Add the Kafka topic properties you need to customize to this table. For further information about the topic-level properties you can define in this table, see the section describing the topic-level configuration in Kafka’s documentation in <a href="http://kafka.apache.org/documentation.html#topic-config">http://kafka.apache.org/documentation.html#topic-config</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used standalone to create a topic that the other Kafka components can use in the other Jobs or Subjobs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the <strong>Install</strong> button on the <strong>Component</strong> tab view. You can also find out and add all missing JARs easily on the <strong>Modules</strong> tab in the <strong>Integration</strong> perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

Related scenarios

No scenario is available for the Standard version of this component yet.
tKafkaInput

Transmits messages you need to process to the components that follow in the Job you are designing. tKafkaInput is a generic message broker that transmits messages to the Job that runs transformations over these messages.

**tKafkaInput Standard properties**

These properties are used to configure tKafkaInput running in the Standard Job framework.

The Standard tKafkaInput component belongs to the Internet family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
Note that the schema of this component is read-only. It stores the messages sent from the message producer. |
|----------------------------|--------------------------------------------------------------------------------------------------|
| **Output type**            | Select the type of the data to be sent to the next component.
Typically, using **String** is recommended, because tKafkaInput can automatically translate the Kafka byte[] messages into strings to be processed by the Job. However, in case that the format of Kafka messages is not known to tKafkaInput, such as Protobuf, you can select byte[] and then use a Custom code component such as tJavaRow to deserialize the messages into strings so that the other components of the same Job can process these messages. |
| **Use an existing connection** | Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined. |
| **Version**                | Select the version of the Kafka cluster to be used. |
| **Zookeeper quorum list**  | Enter the address of the Zookeeper service of the Kafka cluster to be used.
The form of this address should be hostname:port. This information is the name and the port of the hosting node in this Kafka cluster.
If you need to specify several addresses, separate them using a comma (,).
This field is available to Kafka 0.8.2.0 only. |
| **Broker list**          | Enter the addresses of the broker nodes of the Kafka cluster to be used.  
|                         | The form of this address should be *hostname:port*. This information is the name and the port of the hosting node in this Kafka cluster.  
|                         | If you need to specify several addresses, separate them using a comma (,).  
|                         | This field is available since Kafka 0.9.0.1. |
| **Reset offsets on consumer group** | Select this check box to clear the offsets saved for the consumer group to be used so that this consumer group is handled as a new group that has not consumed any messages. |
| **New consumer group starts from** | Select the starting point from which the messages of a topic are consumed.  
|                         | In Kafka, the increasing ID number of a message is called *offset*. When a new consumer group starts, from this list, you can select **beginning** to start consumption from the oldest message of the entire topic, or select **latest** to wait for a new message.  
|                         | Note that the consumer group takes into account only the offset-committed messages to start from.  
|                         | Each consumer group has its own counter to remember the position of a message it has consumed. For this reason, once a consumer group starts to consume messages of a given topic, a consumer group recognizes the latest message only with regard to the position where this group stops the consumption, rather than to the entire topic. Based on this principle, the following behaviors can be expected:  
|                         | • If you are resuming an existing consumer group, this option determines the starting point for this consumer group only if it does not already have a committed starting point. Otherwise, this consumer group starts from this committed starting point. For example, a topic has 100 messages. If an existing consumer group has successfully processed 50 messages, and has committed their offsets, then the same consumer group restarts from the offset 51.  
|                         | • If you create a new consumer group or reset an existing consumer group, which, in either case, means this group has not consumed any message of this topic, then when you start it from latest, this new group starts and waits for the offset 101. |
| **Offset storage** | Select the system to which you want to commit the offsets of the consumed messages. |
| **Enable dual commit** | If you select **Kafka** as the offset storage system, the **Enable dual commit** check box is displayed. By default it is selected to let the Job commit the messages to both Zookeeper and Kafka. If you want the Job to commit only to Kafka, clear this check box. |
Auto-commit offsets | Select this check box to make **tKafkaInput** automatically save its consumption state at the end of each given time interval. You need to define this interval in the **Interval** field that is displayed. Note that the offsets are committed only at the end of each interval. If your Job stops in the middle of an interval, the message consumption state within this interval is not committed.

Topic name | Enter the name of the topic from which **tKafkaInput** receives the feed of messages.

Consumer group ID | Enter the name of the consumer group to which you want the current consumer (the **tKafkaInput** component) to belong. This consumer group will be created at runtime if it does not exist at that moment.

Stop after a maximum total duration (ms) | Select this check box and in the pop-up field, enter the duration (in milliseconds) at the end of which **tKafkaInput** stops running.

Stop after receiving a maximum number of messages | Select this check box and in the pop-up field, enter the maximum number of messages you want **tKafkaInput** to receive before it automatically stops running.

Stop after maximum time waiting between messages (ms) | Select this check box and in the pop-up field, enter the waiting time (in milliseconds) by **tKafkaInput** for a new message. If **tKafkaInput** does not receive any new message when this waiting time meets its end, it automatically stops running.

Use SSL/TLS | Select this check box to enable the SSL or TLS encrypted connection. Then you need to use the **tSetKeystore** component in the same Job to specify the encryption information. For further information about **tSetKeystore**, see **tSetKeystore** on page 3532. This check box is available since Kafka 0.9.0.1.

Use Kerberos authentication | If the Kafka cluster to be used is secured with Kerberos, select this check box to display the related parameters to be defined:

- **JAAS configuration path**: enter the path, or browse to the JAAS configuration file to be used by the Job to authenticate as a client to Kafka.

  This JAAS file describes how the clients, the Kafka-related Jobs in terms of **Talend**, can connect to the Kafka broker nodes, using either the kinit mode or the keytab mode. It must be stored in the machine where these Jobs are executed.

  **Talend**, Kerberos or Kafka does not provide this JAAS file. You need to create it by following the explanation in **Configuring Kafka client** depending on the security strategy of your organization.
- **Kafka brokers principal name**: enter the primary part of the Kerberos principal you defined for the brokers when you were creating the broker cluster. For example, in this principal `kafka/kafka1.hostname.com@example.com`, the primary part to be used to fill in this field is `kafka`.

- **Set kinit command path**: Kerberos uses a default path to its kinit executable. If you have changed this path, select this check box and enter the custom access path. If you leave this check box clear, the default path is used.

- **Set Kerberos configuration path**: Kerberos uses a default path to its configuration file, the `krb5.conf` file (or `krb5.ini` in Windows) for Kerberos 5 for example. If you have changed this path, select this check box and enter the custom access path to the Kerberos configuration file. If you leave this check box clear, a given strategy is applied by Kerberos to attempt to find the configuration information it requires. For details about this strategy, see the Locating the krb5.conf Configuration File section in Kerberos requirements.

For further information about how a Kafka cluster is secured with Kerberos, see Authenticating using SASL. This check box is available since Kafka 0.9.0.1.

### Advanced settings

<table>
<thead>
<tr>
<th>Kafka properties</th>
<th>Add the Kafka consumer properties you need to customize to this table. For example, you can set a specific <code>zookeeper.connection.timeout.ms</code> value to avoid <code>ZkTimeoutException</code>. For further information about the consumer properties you can define in this table, see the section describing the consumer configuration in Kafka’s documentation in <a href="http://kafka.apache.org/documentation.html">http://kafka.apache.org/documentation.html</a>#consumerconfigs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout precision(ms)</td>
<td>Enter the time duration in millisecond at the end of which you want a timeout exception to be returned if no message is available for consumption. The value -1 indicates that no timeout is set.</td>
</tr>
<tr>
<td>Load the offset with the message</td>
<td>Select this check box to output the offsets of the consumed messages to the next component. When selecting it, a read-only column called <code>offset</code> is added to the schema.</td>
</tr>
<tr>
<td>Custom encoding</td>
<td>You may encounter encoding issues when you process the stored data. In that situation, select this check box to display the <strong>Encoding</strong> list.</td>
</tr>
</tbody>
</table>
Select the encoding from the list or select Custom and define it manually.

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

**Usage**

| Usage rule | This component is used as a start component and requires an output link. When the Kafka topic it needs to use does not exist, it can be used along with the tKafkaCreateTopic component to read the topic created by the latter component. |

**Related scenarios**

No scenario is available for the Standard version of this component yet.
**tKafkaOutput**

Publishes messages into a Kafka system.

This component receives messages serialized into byte arrays by its preceding component and issues these messages into a given Kafka system.

**tKafkaOutput Standard properties**

These properties are used to configure tKafkaOutput running in the Standard Job framework.

The Standard tKafkaOutput component belongs to the Internet family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields. Note that the schema of this component is read-only. It stores the messages to be published.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>Select the version of the Kafka cluster to be used.</td>
</tr>
<tr>
<td><strong>Broker list</strong></td>
<td>Enter the addresses of the broker nodes of the Kafka cluster to be used. The form of this address should be <code>hostname:port</code>. This information is the name and the port of the hosting node in this Kafka cluster. If you need to specify several addresses, separate them using a comma (,).</td>
</tr>
<tr>
<td><strong>Topic name</strong></td>
<td>Enter the name of the topic you want to publish messages to. This topic must already exist.</td>
</tr>
<tr>
<td><strong>Compress the data</strong></td>
<td>Select the Compress the data check box to compress the output data.</td>
</tr>
<tr>
<td><strong>Use SSL/TLS</strong></td>
<td>Select this check box to enable the SSL or TLS encrypted connection. Then you need to use the tSetKeystore component in the same Job to specify the encryption information. For further information about tSetKeystore, see tSetKeystore on page 3532. This check box is available since Kafka 0.9.0.1.</td>
</tr>
</tbody>
</table>
Use Kerberos authentication

If the Kafka cluster to be used is secured with Kerberos, select this check box to display the related parameters to be defined:

- **JAAS configuration path**: enter the path, or browse to the JAAS configuration file to be used by the Job to authenticate as a client to Kafka.

  This JAAS file describes how the clients, the Kafka-related Jobs in terms of Talend, can connect to the Kafka broker nodes, using either the kinit mode or the keytab mode. It must be stored in the machine where these Jobs are executed.

  Talend, Kerberos or Kafka does not provide this JAAS file. You need to create it by following the explanation in Configuring Kafka client depending on the security strategy of your organization.

- **Kafka brokers principal name**: enter the primary part of the Kerberos principal you defined for the brokers when you were creating the broker cluster. For example, in this principal `kafka/kafka1.hostname.com@EXAMPLE.COM`, the primary part to be used to fill in this field is `kafka`.

- **Set kinit command path**: Kerberos uses a default path to its kinit executable. If you have changed this path, select this check box and enter the custom access path.

  If you leave this check box clear, the default path is used.

- **Set Kerberos configuration path**: Kerberos uses a default path to its configuration file, the `krb5.conf` file (or `krb5.ini` in Windows) for Kerberos 5 for example.

  If you have changed this path, select this check box and enter the custom access path to the Kerberos configuration file.

  If you leave this check box clear, a given strategy is applied by Kerberos to attempt to find the configuration information it requires. For details about this strategy, see the Locating the krb5.conf Configuration File section in Kerberos requirements.

For further information about how a Kafka cluster is secured with Kerberos, see Authenticating using SASL.

This check box is available since Kafka 0.9.0.1.

---

### Advanced settings

<table>
<thead>
<tr>
<th>Kafka properties</th>
<th>Add the Kafka new producer properties you need to customize to this table.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For further information about the new producer properties you can define in this table, see the section describing the new producer configuration in Kafka’s documentation in <a href="http://kafka.apache.org/documentation.html#newproducerconfigs">http://kafka.apache.org/documentation.html#newproducerconfigs</a>.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the processing metadata at the Job level as well as at each component level.</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

**Usage**

**Usage rule**

This component is an end component. It requires a `tJavaRow` or `tJava` component to transform the incoming data into serialized byte arrays.

The following sample shows how to construct a statement to perform this transformation:

```java
output_rowserializedValue = input_row.users.getBytes();
```

In this code, the `output_row` variable represents the schema of the data to be output to `tKafkaOutput` and `output_rowserializedValue` the single read-only column of that schema; the `input_row` variable represents the schema of the incoming data and `input_row.users` the input column called `users` to be transformed to byte arrays by the `getBytes()` method.

**Related scenarios**

No scenario is available for the Standard version of this component yet.
**tLDAPAttributesInput**

Analyses each object found via the LDAP query and lists a collection of attributes associated with the object.

tLDAPAttributesInput executes an LDAP query based on the given filter and corresponding to the schema definition. Then it passes on the field list to the next component via a **Main row** link.

**tLDAPAttributesInput Standard properties**

These properties are used to configure tLDAPAttributesInput running in the Standard Job framework.
The Standard tLDAPAttributesInput component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

| Use an existing connection | Select this check box and click the relevant tLDAPConnection component on the Component list to reuse the connection details you already defined. |

<table>
<thead>
<tr>
<th>Host</th>
<th>LDAP Directory server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of server.</td>
</tr>
<tr>
<td>Base DN</td>
<td>Path to user’s authorised tree leaf.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Select the protocol type on the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>LDAP</strong>: no encryption is used</td>
</tr>
<tr>
<td></td>
<td><strong>LDAPS</strong>: secured LDAP. When this option is chosen, the Advanced CA check box appears. Once selected, the advanced mode allows you to specify the directory and the keystore password of the certificate file for storing a specific CA. However, you can still deactivate this certificate validation by selecting the Trust all certs check box.</td>
</tr>
<tr>
<td></td>
<td><strong>TLS</strong>: certificate is used. When this option is chosen, the Advanced CA check box appears and is used the same way as that of the LDAPS type.</td>
</tr>
</tbody>
</table>

| Authentication User and Password | Select the Authentication check box if LDAP login is required. Note that the login must match the LDAP syntax requirement to be valid. e.g.: “cn=Directory Manager”. |
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

<table>
<thead>
<tr>
<th>Filter</th>
<th>Type in the filter as expected by the LDAP directory db.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi valued field separator</td>
<td>Type in the value separator in multi-value fields.</td>
</tr>
<tr>
<td><strong>Alias dereferencing</strong></td>
<td>Select the option on the list. Never improves search performance if you are sure that no alias is to be dereferenced. By default, Always is to be used:</td>
</tr>
<tr>
<td>Always</td>
<td>Always dereference aliases</td>
</tr>
<tr>
<td>Never</td>
<td>Never dereferences aliases</td>
</tr>
<tr>
<td>Searching</td>
<td>Dereferences aliases only after name resolution.</td>
</tr>
<tr>
<td>Finding</td>
<td>Dereferences aliases only during name resolution.</td>
</tr>
</tbody>
</table>

| **Referral handling**       | Select the option on the list: |
| Ignored                     | does not handle request redirections |
| Follow                      | does handle request redirections |

| **Limit**                   | Fill in a limit number of records to be read if needed. |
| **Time Limit**              | Fill in a timeout period for the directory. access |
| **Paging**                  | Specify the number of entries returned at a time by the LDAP server. |

| **Die on error**            | This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row > Rejects link. |

| **Schema and Edit Schema**  | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. |

| Built-in                    | The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide. |
| Repository                  | The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide. |

**Warning:**
As this component is intended to list the attributes associated with a LDAP object, its schema is then pre-defined. You should retain these established columns, even though you may need to add some new columns. Hence you should use the Built-in mode.
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

The pre-defined schema lists:
- **objectclass**: list of object classes
- **mandatoryattributes**: list of mandatory attributes to these classes
- **optionalattributes**: list of optional attributes to these classes
- **objectattributes**: list of attributes that are essential for the analysed object.

### Advanced settings

<table>
<thead>
<tr>
<th>Class Definition Root</th>
<th>Specify the root of the object class definition namespace.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the job processing metadata at a job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

| Global Variables | **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
|------------------|------------------------------------------------------------|
|                  | **RESULT_NAME**: the name of the current LDAP entry satisfying the search filter. This is a flow variable, and it returns a string.  
|                  | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
|                  | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
|                  | To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. |
For further information about variables, see *Talend Studio User Guide*.

### Usage

#### Usage rule

This component covers all possible LDAP queries.  
Note: Press *Ctrl + Space bar* to access the global variable list, including the *GetResultName* variable to retrieve automatically the relevant Base

### Related scenario

The *tLDAPAttributesInput* component follows the usage similar to that of *tLDAPInput*. Hence for *tLDAPInput* related scenario, see [Scenario: Displaying LDAP directory’s filtered content](#) on page 1957.
tLDAPClose

Disconnects one connection to the LDAP Directory server so as to release occupied resources.

## tLDAPClose Standard properties

These properties are used to configure tLDAPClose running in the Standard Job framework.
The Standard tLDAPClose component belongs to the Databases family.
The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Component list</th>
<th>Select the tLDAPConnection component in the list if more than one connection is planned for the current Job.</th>
</tr>
</thead>
</table>

### Advanced settings

- **tStat Catcher Statistics**
  - Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Usage

- **Usage rule**
  - This component is to be used along with other LDAP components, especially with tLDAPConnection.

- **Dynamic settings**
  - Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

  Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

  For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

  For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Related scenarios

No scenario is available for the Standard version of this component yet.
tLDAPConnection

Creates a connection to an LDAP Directory server.

tLDAPConnection can be invoked by other components that need to access the LDAP Directory server, e.g., tLDAPInput, tLDAPOutput, etc.

**tLDAPConnection Standard properties**

These properties are used to configure tLDAPConnection running in the Standard Job framework.
The Standard tLDAPConnection component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Host**
LDAP Directory server IP address.

**Port**
Listening port number of server.

**Protocol**
Select the protocol type on the list.

- **LDAP**: no encryption is used
- **LDAPS**: secured LDAP. When this option is chosen, the **Advanced CA** check box appears. Once selected, the advanced mode allows you to specify the directory and the keystore password of the certificate file for storing a specific CA. However, you can still deactivate this certificate validation by selecting the **Trust all certs** check box.
- **TLS**: certificate is used. When this option is chosen, the **Advanced CA** check box appears and is used the same way as that of the **LDAPS** type.

**Base DN**
Path to user’s authorized tree leaf.

**User and Password**
Fill in the **User** and **Password** as required by the directory

Note that the login must match the LDAP syntax requirement to be valid. e.g.: ‘cn=Directory Manager’.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.
### Alias dereferencing

Select the option on the list. **Never** improves search performance if you are sure that no aliases is to be dereferenced. By default, **Always** is to be used:

- **Always**: Always dereference aliases
- **Never**: Never dereferences aliases.
- **Searching**: Dereferences aliases only after name resolution.
- **Finding**: Dereferences aliases only during name resolution

### Referral handling

Select the option on the list:

- **Ignore**: does not handle request redirections
- **Follow**: does handle request redirections

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to gather the job processing metadata at a job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component is to be used with other LDAP components, especially with \texttt{tLDAPInput} and \texttt{tLDAPOutput}.</td>
</tr>
</tbody>
</table>

### Related scenarios

This component is closely related to \texttt{tLDAPInput} and \texttt{tLDAPOutput} as it frees you from filling in the connection details repeatedly if multiple LDAP input/output components exist.

For \texttt{tLDAPConnection} related scenarios, see \textit{Scenario: Inserting data in mother/daughter tables} on page 2476.
tLDAPInput

Executes an LDAP query based on the given filter and corresponding to the schema definition. Then it passes on the field list to the next component via a **Main row** link.

### tLDAPInput Standard properties

These properties are used to configure tLDAPInput running in the Standard Job framework.

The Standard tLDAPInput component belongs to the Databases family.

The component in this framework is generally available.

#### Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see **Talend Studio User Guide**.

#### Use an existing connection

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

Note that when a Job contains the parent Job and the child Job, **Component List** presents only the connection components in the same Job level.

| Host | LDAP Directory server IP address. |
| Port | Listening port number of server. |
| Base DN | Path to the user’s authorised tree leaf. |

**Note:**
To retrieve the full DN information, enter a field named **DN** in the schema, in either upper case or lower case.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Select the protocol type on the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP</td>
<td>no encryption is used</td>
</tr>
<tr>
<td>LDAPS</td>
<td>secured LDAP. When this option is chosen, the <strong>Advanced CA</strong> check box appears. Once selected, the advanced mode allows you to specify the directory and the keystore password of the certificate file for storing</td>
</tr>
</tbody>
</table>
a specific CA. However, you can still deactivate this certificate validation by selecting the **Trust all certs** check box.

**TLS**: certificate is used. When this option is chosen, the **Advanced CA** check box appears and is used the same way as that of the **LDAPS** type.

| **Authentication User** and **Password** | Select the **Authentication** check box if LDAP login is required. Note that the login must match the LDAP syntax requirement to be valid. E.g.: “cn=Directory Manager”.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
|----------------------------------------|

| **Filter** | Type in the filter as expected by the LDAP directory db. |
|----------------|

| **Multi valued field separator** | Type in the value separator in multi-value fields. |
|----------------|

| **Alias dereferencing** | Select the option on the list. **Never** improves search performance if you are sure that no alias is to be dereferenced. By default, **Always** is to be used:

**Always**: Always dereference aliases

**Never**: Never dereferences aliases.

**Searching**: Dereferences aliases only after name resolution.

**Finding**: Dereferences aliases only during name resolution |
|----------------|

| **Referral handling** | Select the option on the list:

**Ignore**: does not handle request redirections

**Follow**: does handle request redirections |
|----------------|

| **Limit** | Fill in a limit number of records to be read if needed. |
|----------------|

| **Time Limit** | Fill in a timeout period for the directory access. |
|----------------|

| **Paging** | Specify the number of entries returned at a time by the LDAP server. |
|----------------|

| **Die on error** | This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link. |
|----------------|

| **Schema** and **Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available: |
|----------------|
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Warning:**
Only three data types are supported here: String, byte[], and List. **tMap** can be used for data type conversion if needed.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see **Talend Studio User Guide**.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see **Talend Studio User Guide**.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RESULT_NAME: the name of the current LDAP entry satisfying the search filter. This is a flow variable, and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible LDAP queries.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note: Press Ctrl + Space bar to access the global variable list, including the GetResultName variable to retrieve automatically the relevant Base.</td>
</tr>
</tbody>
</table>
Scenario: Displaying LDAP directory’s filtered content

The Job described below simply filters the LDAP directory and displays the result on the console.

Adding and linking the components

Procedure

Drop the **tLDAPInput** component along with a **tLogRow** from the Palette to the design workspace.

Configuring the components

Procedure

1. Set the **tLDAPInput** properties.
2. Set the **Property type** on **Repository** if you stored the LDAP connection details in the **Metadata Manager** in the **Repository**. Then select the relevant entry on the list.
3. In **Built-In** mode, fill in the **Host** and **Port** information manually. Host can be the IP address of the LDAP directory server or its DNS name.
4. No particular **Base DN** is to be set.

5. Then select the relevant **Protocol** on the list. In this example: a simple **LDAP** protocol is used.
6. Select the **Authentication** check box and fill in the login information if required to read the directory. In this use case, no authentication is needed.

7. In the **Filter** area, type in the command, the data selection is based on. In this example, the filter is:
   
   \((\&(objectClass=inetorgperson)(uid=PIERRE DUPONT))\).

8. Fill in **Multi-valued field separator** with a comma as some fields may hold more than one value, separated by a comma.

9. As we do not know if some aliases are used in the LDAP directory, select **Always** on the list.

10. Set **Ignore** as **Referral handling**.

11. Set the limit to **100** for this use case.

12. Set the **Schema** as required by your LDAP directory. In this example, the schema is made of 6 columns including the `objectClass` and `uid` columns which get filtered on.

<table>
<thead>
<tr>
<th>Column</th>
<th>Db Column</th>
<th>Key</th>
<th>Type</th>
<th>Nullable</th>
<th>Date P.</th>
<th>Length</th>
<th>Pre.</th>
</tr>
</thead>
<tbody>
<tr>
<td>dc</td>
<td>dc</td>
<td></td>
<td>string</td>
<td>✓</td>
<td></td>
<td>255</td>
<td></td>
</tr>
<tr>
<td>ou</td>
<td>ou</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
<td>255</td>
<td></td>
</tr>
<tr>
<td>objectClass</td>
<td>objectClass</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
<td>255</td>
<td></td>
</tr>
<tr>
<td>mail</td>
<td>mail</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
<td>255</td>
<td></td>
</tr>
<tr>
<td>uid</td>
<td>uid</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
<td>255</td>
<td></td>
</tr>
<tr>
<td>dn</td>
<td>dn</td>
<td></td>
<td>String</td>
<td>[ ]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13. In the **tLogRow** component, no particular setting is required.

**Results**


| DATA| top, person, organizationalPerson, inetorgperson, s4100User| nhirt78@talend.com, PIERRE DUPONT |

Job testLDAPInput ended at 16:05 18/09/2007. [exit code=0]

Only one entry of the directory corresponds to the filter criteria given in the **tLDAPInput** component.
tLDAPOutput

Executes an LDAP query based on the given filter and corresponding to the schema definition. Then it passes on the field list to the next component via a Main row link.

**tLDAPOutput Standard properties**

These properties are used to configure tLDAPOutput running in the Standard Job framework.
The Standard tLDAPOutput component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

![Open database connection wizard](image)

Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.

<table>
<thead>
<tr>
<th>Host</th>
<th>LDAP Directory server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of server.</td>
</tr>
<tr>
<td>Base DN</td>
<td>Path to user’s authorized tree leaf.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Select the protocol type on the list.</td>
</tr>
<tr>
<td></td>
<td>LDAP : no encryption is used</td>
</tr>
<tr>
<td></td>
<td>LDAPS: secured LDAP. When this option is chosen, the Advanced CA check box appears. Once selected, the advanced mode allows you to specify the directory and the keystore password of the certificate file for storing a specific CA. However, you can still deactivate this certificate validation by selecting the Trust all certs check box.</td>
</tr>
</tbody>
</table>
**TLS**: certificate is used When this option is chosen, the **Advanced CA** check box appears and is used the same way as that of the **LDAPS** type.

### User and Password

Fill in the **User** and **Password** as required by the directory.

Note that the login must match the LDAP syntax requirement to be valid. e.g.: "cn=Directory Manager".

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

### Multi valued field separator

Type in the value separator in multi-value fields.

### Alias dereferencing

Select the option on the list. **Never** improves search performance if you are sure that no aliases is to be dereferenced. By default, **Always** is to be used:

- **Always**: Always dereference aliases
- **Never**: Never dereferences aliases.
- **Searching**: Dereferences aliases only after name resolution.
- **Finding**: Dereferences aliases only during name resolution

### Referral handling

Select the option on the list:

- **Ignore**: does not handle request redirections
- **Follow**: does handle request redirections

### Insert mode

Select the editing mode on the list:

- **Add**: add a value in a multi-value attribute,
- **Insert**: insert new data,
- **Update**: updates the existing data,
- **Delete**: remove the selected data from the directory,
- **Insert or update**: Insert a new record. If the record with the given reference already exists, an update would be made.

**Warning:**

When **Update** is selected, you must go to the **Advanced settings** view, and select the **Use Attribute Options (for update mode)** check box to show the **Attribute Options** table. Then, select the attribute to update under the **Attribute name** part and the choose **Replace** under the **Option** part.

### DN Column Name

Select in the list the type of the LDAP input entity used.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-In** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Die on error**

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Reject** link.

**Advanced settings**

**Use Attribute Options (for update mode)**

Select this check box to choose the desired attribute (including dn, dc, ou, objectClass, mail and uid) and the corresponding operation (including Add, Replace, Remove Attribute and Remove Value).

**tStatCatcher Statistics**

Select this check box to gather the job processing metadata at a job level as well as at each component level.

**Global Variables**

**Global Variables**

**NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

**NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer.
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | This component covers all possible LDAP queries. | Note: Press **Ctrl + Space bar** to access the global variable list, including the **GetResultName** variable to retrieve the relevant DN Base automatically. This component allows you to carry out actions on a table or on the data of a table in a database. It also allows you to create a reject flow using a **Row > Rejects** link to filter data in error. For an example of **tMySqlOutput** in use, see *Scenario: Retrieving data in error with a Reject link* on page 2529. |

### Scenario: Editing data in a LDAP directory

The following scenario describes a Job that reads an LDAP directory, updates the email of a selected entry and displays the output before writing the LDAP directory. To keep it simple, no alias dereferencing nor referral handling is performed. This scenario is based on *Scenario: Displaying LDAP directory’s filtered content* on page 1957. The result returned was a single entry, related to an organisational person, whom email is to be updated.

The output shows the following fields: **dn**, **uid** and **mail** as defined in the Job.

### Dropping the components

**Procedure**

1. Drop the **tLDAPInput**, **tLDAPOutput**, **tMap** and **tLogRow** components from the **Palette** to the design workspace.
2. Link **tLDAPInput** to **tMap** using the **Row > Main** connection.
3. Link **tMap** to **tLogRow** using the **Row > Main** connection.
4. Link **tLogRow** to **tLDAPOutput** using the **Row > Main** connection.
Configuring the components

Procedure

1. In the tLDAPInput Component view, set the connection details to the LDAP directory server as well as the filter as described in Scenario: Displaying LDAP directory's filtered content on page 1957. Change the schema to make it simpler, by removing the unused fields: `dc`, `ou`, `objectclass`.

2. Then open the mapper to set the edit to be carried out.
   Drag & drop the `uid` column from the input table to the output as no change is required on this column.

3. In the Expression field of the `dn` column (output), fill in with the exact expression expected by the LDAP server to reach the target tree leaf and allow directory writing only if you haven't set it already in the Base DN field of the tLDAPOutput component.

4. In this use case, the GetResultName global variable is used to retrieve this path automatically. Press Ctrl+Space bar to access the variable list and select tLDAPInput_1_RESULT_NAME.

5. In the `mail` column's expression field, type in the new email that will overwrite the current data in the LDAP directory. In this example, we change to Pierre.Dupont@talend.com. Click OK to validate the changes.

6. Then select the tLDAPOutput component to set the directory writing properties.
7. Set the Port and Host details manually if they aren't stored in the Repository.

8. In Base DN field, set the highest tree leaf you have the rights to access. If you have not set previously the exact and full path of the target DN you want to access, then fill in it here. In this use case, the full DN is provided by the \textit{dn} output from the \texttt{tMap} component, therefore only the highest accessible leaf is given: \texttt{o=directoryRoot}.

9. Select the relevant protocol to be used: LDAP for this example. Fill in the User and Password as expected by the LDAP directory. Fill in \textit{Multi-valued field separator} with a comma as some fields may hold more than one value, separated by a comma.

10. Use the default setting of Aliases Dereferencing and Referral Handling fields, respectively Always and Ignore.

The Insert mode for this use case is Update (the email address). The schema was provided by the previous component through the propagation operation.

11. In the Advanced settings view, select the Use Attribute Options (for update mode) check box to show the Attribute Options table. Select the attribute mail under the Attribute Name part and the choose Replace under the Option part.
Executing the Job

Procedure

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to run the Job.

Starting job LDAPInputnew at 14.17 20/09/2007
uid=PIERRE DUPONT,ou=DATA,o=TALEND,c=TALEND|PIERRE DUPONT|Pierre.Dupont@talend.com
Job LDAPInputnew ended at 14.17 20/09/2007. [exit code=0]
tLDAPRenameEntry

Renames ones or more entries in a specific LDAP directory.

**tLDAPRenameEntry Standard properties**

These properties are used to configure tLDAPRenameEntry running in the Standard Job framework. The Standard tLDAPRenameEntry component belongs to the Databases family. The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and click the relevant tLDAPConnection component on the Component list to reuse the connection details you already defined.

**Host**

LDAP directory server IP address.

**Port**

Number of the listening port of the server.

**Base DN**

Path to user’s authorized tree leaf.

**Protocol**

Select the protocol type on the list.

- **LDAP**: no encryption is used,
- **LDAPS**: secured LDAP. When this option is chosen, the Advanced CA check box appears. Once selected, the advanced mode allows you to specify the directory and the keystore password of the certificate file for storing a specific CA. However, you can still deactivate this certificate validation by selecting the Trust all certs check box.
- **TLS**: certificate is used. When this option is chosen, the Advanced CA check box appears and is used the same way as that of the LDAPS type.

**User and Password**

Fill in user authentication information.

Note that the login must match the LDAP syntax requirement to be valid. e.g.: "cn=Directory Manager".

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.
| Alias dereferencing | Select the option on the list. **Never** improves search performance if you are sure that no alias is to be dereferenced. By default, **Always** is to be used:

**Always**: Always dereference aliases,

**Never**: Never dereferences aliases,

**Searching**: Dereferences aliases only after name resolution,

**Finding**: Dereferences aliases only during name resolution. |
| --- | --- |
| Referrals handling | Select the option on the list:

- **Ignore**: does not handle request redirections,
- **Follow**: does handle request redirections. |
| Previous DN and New DN | Select from the list the schema column that holds the old DN (**Previous DN**) and the column that holds the new DN (**New DN**). |
| Schema and Edit Schema | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window. |
| Built-in | The schema is created and stored locally for this component only. Related topic: see **Talend Studio User Guide**. |
| Repository | The schema already exists and is stored in the Repository, hence can be reused. Related topic: see **Talend Studio User Guide**. |
| Die on error | This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Reject** link. |
Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

Global Variables

| Global Variables | NB_LINE: the number of rows processed. This is an After variable and it returns an integer. |
| | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |
| | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
| | To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. |
| | For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component covers all possible LDAP queries. It is usually used as a one-component subjob but you can use it with other components as well. |
| | Note: Press Ctrl + Space bar to access the global variable list, including the GetResultName variable to retrieve automatically the relevant DN Base. |

Related scenarios

For use cases in relation with tLDAPRenameEntry, see the following scenarios:

- Scenario: Displaying LDAP directory’s filtered content on page 1957.
- Scenario: Editing data in a LDAP directory on page 1962.
tLibraryLoad

Loads useable Java libraries in a Job.

**tLibraryLoad Standard properties**

These properties are used to configure tLibraryLoad running in the Standard Job framework.

The Standard tLibraryLoad component belongs to the Custom Code family.

The component in this framework is generally available.

**Basic settings**

| Library | Select the library you want to import from the list, or click on the [...] button to browse to the library in your directory. |

**Advanced settings**

| Dynamic Libs | **Lib Paths**: Enter the access path to your library, between double quotation marks. |

| Import | Enter the Java code required to import, if required, the external library used in the code editing field of the Basic settings tab of the components such as tJava or tJavaFlex. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. |

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. |

For further information about variables, see *Talend Studio User Guide*. |

**Usage**

| Usage rule | This component may be used alone, although it is more logical to use it as part of a Job. |

| Limitation | The library is loaded locally. |
Scenario: Checking the format of an e-mail address

This scenario uses two components, a tLibraryLoad and a tJava. The goal of this scenario is to check the format of an e-mail address and verify whether the format is valid or not.

Setting up the Job

Procedure

1. In the Palette, open the Custom_Code folder, and slide a tLibraryLoad and tJava component onto the workspace.
2. Connect tLibraryLoad to tJava using a Trigger > OnSubjobOk link.

Configuring the tLibraryLoad component

Procedure

1. Double-click on tLibraryLoad to display its Basic settings. From the Library list, select jakarta-oro-2.0.8.jar.
2. In the Import field of the Advanced settings tab, type import org.apache.oro.text.regex.*;
Configuring the tJava component

Procedure

1. Double-click on tJava to display its Component view.
2. In the Basic settings tab, enter your code, as in the screenshot below. The code allows you to check whether the character string pertains to an e-mail address, based on the regular expression: "^[\w_.-]+@[\w_.-]+\.[\w]+$".

```java
String email="ma@coocooom";

//Validate if the email string is correct or not based on a regexp pattern
Perl5Matcher matcher = new Perl5Matcher();
Perl5Compiler compiler = new Perl5Compiler();
Pattern pattern = compiler.compile("^[\w_.-]+@[\w_.-]+\.[\w]+$");

if (!matcher.matches(email, pattern)) {
    System.out.println(email + " : + false);
} else {
    System.out.println(email + " : + true);
}
```

Job execution

Press F6 to save and run the Job.

```
Starting job LibraryLoad at 16:51 18/02/2010.
[statistics] connecting to socket on port 3979
[statistics] connected
ma@coocooom : false
[statistics] disconnected
Job LibraryLoad ended at 16:51 18/02/2010. (exit code=0)
```

The Console displays the boolean false. Hence, the e-mail address is not valid as the format is incorrect.
tLineChart

Reads data from an input flow and transforms the data into a line chart in a PNG image file to ease technical analysis.

**tLineChart Standard properties**

These properties are used to configure tLineChart running in the Standard Job framework.
The Standard tLineChart component belongs to the Business Intelligence family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>The schema of tLineChart contains three read-only columns named <code>series</code> (string), <code>x</code> (integer), and <code>y</code> (integer) respectively, in a fixed order. The data in any extra columns will be only passed to the next component, if any, without being presented in the generated line chart.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in:</strong> The schema is created and stored locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Sync columns</strong></td>
<td>Click to synchronize the output file schema with the input file schema. The Sync function only displays once the Row connection is linked with the output component.</td>
</tr>
<tr>
<td><strong>Generated image path</strong></td>
<td>Name and path of the output image file.</td>
</tr>
<tr>
<td><strong>Chart title</strong></td>
<td>Enter the title of the line chart to be generated.</td>
</tr>
<tr>
<td><strong>Domain axis label</strong> and <strong>Range axis label</strong></td>
<td>Enter the domain axis (X axis) and range axis (Y axis) labels.</td>
</tr>
<tr>
<td><strong>Plot orientation</strong></td>
<td>Select the plot orientation of the range axis: <em>Vertical</em> or <em>Horizontal</em>.</td>
</tr>
<tr>
<td><strong>Include legend</strong></td>
<td>Select this check box if you want your line chart to include a legend, indicating the lines of different series in different colors.</td>
</tr>
</tbody>
</table>
**Image width and Image height**
Enter the width and height of the image, in pixels.

**Moving average**
Select this check box to show a moving average for each series on your line chart. With this check box selected, the **Period** field appears, letting you define a period of which you want to show the moving average.

**Lower bound and Upper bound**
Define the lowest and highest values to be displayed on the range axis.

**Chart background and Plot background**
Select the chart background color and the plot area background color.

### Advanced settings

**tStat Catcher Statistics**
Select this check box to collect log data at the component level.

### Global Variables

**Global Variables**

- **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

**Usage rule**
This component is mainly used as Output component. It requires an Input component and Row main link as input.

### Scenario: Creating a line chart to ease trend analysis

This scenario describes a Job that reads data from a CSV file and transforms the data into a line chart to facilitate trend analysis. The input file records how long (in minutes) per week a person watches different TV channels over ten weeks, as shown below:

```
Week;TV_A;TV_B;TV_C
1;327;286;244
```
Because the input file has a different structure than required by the tLineChart component, this use case uses the tMap component to adapt the source data to the three-column schema of tLineChart so that a temporary CSV file can be created as the input to the tLineChart component.

Note:
You will usually use the tMap component to adjust the input schema in accordance with the schema structure of the tLineChart component. For more information about how to use the tMap component, see Talend Studio User Guide and tMap on page 1995.

To ensure correct generation of the temporary input file, a pre-treatment subjob is used to delete the temporary file in case it already exists before the main Job is executed; as this temporary file serves this specific Job only, a post-treatment subjob is used to deleted it after the main Job is executed.

Dropping and linking components

Procedure

1. Drop the following components from the Palette to the design workspace: two tFileDelete components, two tFileInputDelimited components, a tMap, three tFileOutputDelimited components, and a tLineChart.
2. Connect the first tFileInputDelimited to the tMap component using a Row > Main connection.
3. Connect the tMap component to the first tFileOutputDelimited component using a Row > Main connection, and name the connection TV_A.
4. Repeat the step above to connect the tMap component to the other two tFileOutputDelimited components using Row > Main connections, and name the connections TV_B and TV_C respectively.
5. Connect the section tFileInputDelimited to the tLineChart component using a Row > Main connection. When questioned whether to get the schema from the target component, click Yes.
6. Connect the first tFileInputDelimited component to the second tFileInputDelimited component using a Trigger > On Subjob Ok connection.
7. Connect the first tFileDelete component to the first tFileInputDelimited component, and then the second tFileInputDelimited component to the second tFileDelete component, using Trigger > On Subjob Ok connections.
8. Relabel the components to best describe their functionality.
Results

Reading the source data

Procedure

1. Double-click the first `tFileInputDelimited` component, which is labelled `Source_data`, to display its Basic settings view.

2. Fill in the File name field by browsing to the input file.

3. Specify the header row. In this use case, the first row of the input file is the header row. And leave the other parameters as they are.
4. Click **Edit schema** to describe the data structure of the input file. In this use case, the input schema is made of four columns: *Week*, *Mins_TVA*, *Mins_TVB*, and *Mins_TVC*. Upon defining the column names and data type, click **OK** to close the schema dialog box.

![Schema of Source_data](image)

**Adapting the source data to the tLineChart schema**

**Procedure**

1. Double-click the **tMap** to open the **Map Editor**.
   
   You can see an input table on the input panel, *row1* in this example, and three empty output tables, named **TV_A**, **TV_B**, and **TV_C** on the output panel.

2. Use the **Schema editor** to add three columns to each output table: *series* (string), *x* (integer), and *y* (integer).

3. In the relevant **Expression** field of the output tables, enter the text to be presented in the legend area of the line chart, **TV A**, **TV B**, and **TV C** respectively in this example.

4. Drop the *Week* column of the input table onto the *x* column of each output table.

5. Drop the *Mins_TVA* column of the input table onto the *y* column of the **TV_A** table.

6. Drop the *Mins_TVB* column of the input table onto the *y* column of the **TV_B** table.

7. Drop the *Mins_TVC* column of the input table onto the *y* column of the **TV_C** table.
8. Click OK to save the mappings and close the **Map Editor** and propagate the output schemas to the output components.

**Generating the temporary input file**

**Procedure**

1. Double-click the first **tFileOutputDelimited** component to display its **Basic settings** view.

2. In the **File Name** field, define a temporary CSV file to send the mapped data flows to. In this use case, we name this file **Temp.csv**. This file will be used as the input to the **tLineChart** component.

3. Select the **Append** check box.

4. Repeat the steps above to define the properties of the other two **tFileOutputDelimited** components, using exactly the same settings as in the first **tFileOutputDelimited** component.

   **Note:**

   Note that the order of output flows from the **tMap** component is not necessarily the actual order of writing data to the target file. To ensure the target file is correctly generated, we need to delete the file by the same name if it already exists before Job execution and select the **Append** check box in all the **tFileOutputDelimited** components in this step.
Configuring line chart generation

Procedure

1. Double-click the second **tFileInputDelimited** component, which is labelled *Temp_Input*, to display its **Basic settings** view.

2. Fill in the **File name** field with the path to the temporary input file generated by the **tFileOutputDelimited** components. In this use case, the temporary input file to the **tLineChart** is *Temp.csv*.

3. Double-click the **tLineChart** component to display its **Basic settings** view.

4. Click **Edit schema** to open the schema dialog box.
5. Check that the input and output schemas are synchronized. If needed, copy all the columns from the output schema to the input schema by clicking the left-pointing double arrow button. Then, click OK to close the schema dialog box.

6. In the Generated image path field, define the path of the image file to be generated.

7. In the Chart title field, define a title for the line chart. In this use case, enter Average Weekly Viewing (per person) as the chart title.

8. Define the domain (X) and range (Y) axis labels. In this use case, enter Week and Minutes respectively the axis labels.

9. Define the image size, the moving average period, the lower and upper bounds, the chart background color, and the background color of the plot area, as you prefer.

   In this use case, we set the image size to 450 by 450, set the lower and upper bounds to 210 and 340 respectively, select light gray as the chart background color, and keep the rest settings are they are.

**Deleting the temporary file**

**Procedure**

1. Double-click the first tFileDelete component to display its Basic settings view.
2. Fill in the **File name** field with the path to the temporary input file, and clear the **Fail on error** check box to allow the main Job to be executed if the file to delete does not exist.

3. Specify the same file path in the other **tFileDelete** component.

**Executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Press **F6** to launch the Job.

A line chart is generated as defined, showing a graphical comparison of the average weekly viewing time and the viewing trends of different TV channels over the past ten weeks.
tLogCatcher

Operates as a log function triggered by one of the three: Java exception, tDie or tWarn, to collect and transfer log data.

Both the tDie and tWarn components are closely related to the tLogCatcher component. They generally make sense when used alongside a tLogCatcher component in order for the log data collected to be encapsulated and passed on to the output defined.

Fetches set fields and messages from Java Exception, tDie and/or tWarn and passes them on to the next component.

**tLogCatcher Standard properties**

These properties are used to configure tLogCatcher running in the Standard Job framework.

The Standard tLogCatcher component belongs to the Logs & Errors family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description, it defines the fields to be processed and passed on to the next component. The schema of this component is read-only. It describes the properties of the log data. You can click the [...] button next to **Edit schema** to view the predefined schema which contains the following fields:
|                         | • **moment**: the time when the message is caught.
|                         | • **pid**: the process ID of the Job.
|                         | • **root_pid**: the root process ID.
|                         | • **father_pid**: the father process ID.
|                         | • **project**: the name of the project.
|                         | • **job**: the name of the Job.
|                         | • **context**: the context used to run the Job.
|                         | • **priority**: the priority level of the message.
|                         | • **type**: the type of the message.
|                         | • **origin**: the name of the component that triggers the message.
|                         | • **message**: the message content.
|                         | • **code**: the error code level. |
| Catch Java Exception    | Select this check box to trigger the **tCatch** function when a Java Exception occurs in the Job. |
| Catch tDie              | Select this check box to trigger the **tCatch** function when a **tDie** component is called in a Job. |
Catch tWarn

Select this check box to trigger the tCatch function when a tWarn component is called in a Job.

Global Variables

Global Variables

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

This component is usually used as the start component of a secondary Job which is automatically triggered at the end of the main Job.

Catching messages triggered by a tWarn component

This example shows you how to use the tLogCatcher component to catch the messages triggered by a tWarn component.

Creating a Job for catching messages triggered by a tWarn component

Create a Job to trigger some messages using the tWarn component, then catch the messages using the tLogCatcher component and display the messages on the console.
**Procedure**

1. Create a new Job and add a `tRowGenerator` component, a `tWarn` component, a `tLogCatcher` component, and a `tLogRow` component by typing their names in the design workspace or dropping them from the Palette.

2. Link the `tRowGenerator` component to the `tWarn` component using a `Row > Main` connection.

3. Link the `tLogCatcher` component to the `tLogRow` component using a `Row > Main` connection.

**Configuring the Job for catching messages triggered by the `tWarn` component**

Configure the components used in the Job that catches the messages triggered by the `tWarn` component and then displays the messages on the console.

**Procedure**

1. Double-click the `tRowGenerator` component to open its row generator editor.

2. Define the schema by adding one column `id` of Integer type, and select the predefined function `Numeric.sequence(String,int,int)` in the Functions column.

3. Enter the number of records to be generated in the `Number of Rows for RowGenerator` field, 3 in this example. When done, click OK to close the dialog box.

4. Double-click the `tWarn` component to open its Basic settings view.
5. Select Info from the Priority drop-down list.

6. In the Warn message field, enter the message to be triggered when a new record is generated, a new record is generated in this example.

7. Double-click the tLogCatcher component to open its Basic settings view.

8. Select the Catch tWarn check box to catch the messages triggered by the tWarn component.

9. Double-click the tLogRow component to open its Basic settings view, and then select Table (print values in cells of a table) in the Mode area for better readability of the result.

### Executing the Job to catch messages triggered by a tWarn component

After setting up the Job and configuring the components used in the Job for catching messages triggered by the tWarn component, you can then execute the Job and verify the Job execution result.

**Procedure**

1. Press Ctrl + S to save the Job.
2. Press F6 to execute the Job.

As shown above, all messages triggered are collected and displayed on the console.

### Catching the message triggered by a tDie component

This example shows you how to use the tLogCatcher component to catch the message triggered by a tDie component.

### Creating a Job for catching the message triggered by a tDie component

Create a Job to trigger a message using the tDie component, then catch the message using the tLogCatcher component and display the message on the console.
Procedure

1. Create a new Job and add a `tRowGenerator` component, a `tFileOutputDelimited` component, a `tDie` component, a `tJava` component, a `tLogCatcher` component, and a `tLogRow` component by typing their names in the design workspace or dropping them from the Palette.

2. Link the `tRowGenerator` component to the `tFileOutputDelimited` component using a Row > Main connection.

3. Link the `tFileOutputDelimited` component to the `tDie` component using a Trigger > Run if connection.

4. Link the `tRowGenerator` component to the `tJava` component using a Trigger > On Subjob Ok connection.

5. Link the `tLogCatcher` component to the `tLogRow` component using a Row > Main connection.

Configuring the Job for catching the message triggered by the tDie component

Configure the components used in the Job that catches the message triggered by the `tDie` component and then displays the message on the console.

Procedure

1. Double-click the `tRowGenerator` component to open its row generator editor.
2. Define the schema by adding one column `id` of Integer type, and select the predefined function `Numeric.sequence(String,int,int)` in the **Functions** column.

3. Enter the number of rows to be generated in the **Number of Rows for RowGenerator** field, 0 in this example. When done, click **OK** to close the dialog box.

4. Double-click the **tFileOutputDelimited** component to open its **Basic settings** view, and in the **File Name** field, specify the path to the file that will hold the data to be generated.

5. Click the **If** connection, and in the **Condition** field on the **Basic settings** view, specify the condition based on which the **tDie** component will be triggered. In this example, it is `((Integer)globalMap.get("tRowGenerator_1_NB_LINE")) <=0`, which means the **tDie** component will be triggered when the number of rows to be generated is less than or equal to zero.

6. Double-click the **tDie** component to open its **Basic settings** view, and in the **Die message** field, enter the message to be triggered before the Job is killed. In this example, it is **no row generated**.

7. Double-click the **tJava** component to open its **Basic settings** view, and in the **Code** field, enter

   ```java
   System.out.println("The number of rows generated is " + ((Integer)globalMap.get("tRowGenerator_1_NB_LINE")) + ". #This message will not be displayed if no row is generated.");
   ```

Note that this message will be displayed only when the number of rows generated is greater than zero. In this example, the number of rows to be generated is 0, so the Job will be killed and this message will not be displayed.
8. Double-click the **tLogCatcher** component to open its **Basic settings** view and select the **Catch tDie** check box to catch the message triggered by the **tDie** component.

   ![tLogCatcher](image)

   - Basic settings
   - Advanced settings
   - Dynamic settings
   - View
   - Documentation

9. Double-click the **tLogRow** component to open its **Basic settings** view, and then select **Table (print values in cells of a table)** in the **Mode** area for better readability of the result.

**Executing the Job to catch the message triggered by the tDie component**

After setting up the Job and configuring the components used in the Job for catching the message triggered by the **tDie** component, you can then execute the Job and verify the Job execution result.

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to execute the Job.

   ```
   [statistics] connecting to socket on port 3448
   [statistics] connected
   [no row generated]
   tLogRow_1
   moment | pid | root_pid|father_pid|project|job |context|priority|type|origin|message |code |
   -------|--|--|----------|--------|-----|-------|--------|----|-------|--------|-----|
   2016-11-08 12:09:28 | p0V565 | | | | | | | | | |
   [statistics] disconnected
   ```

   As shown above, the message triggered is collected and displayed on the console.

3. Change the number of rows to be generated (10 for example) in the **Number of Rows for RowGenerator** field of the **tRowGenerator** component, save and execute the Job again to find a different result.
tLogRow

Displays data or results in the Run console to monitor data processed.

tLogRow Standard properties

These properties are used to configure tLogRow running in the Standard Job framework.

The Standard tLogRow component belongs to the Logs & Errors family.

The component in this framework is generally available.

Basic settings

| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

| Built-In | You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide. |
| Repository | You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. |

| Sync columns | Click to synchronize the output file schema with the input file schema. The Sync function is available only when the component is linked with the preceding component using a Row connection. |
| Basic | Displays the output flow in basic mode. |
| Table | Displays the output flow in table cells. |
| Vertical | Displays each row of the output flow as a key-value list. |
With this mode selected, you can choose to show either the unique name or the label of component, or both of them, for each output row.

**Separator**  
(For Basic mode only)  
Enter the separator which will delimit data on the Log display.

**Print header**  
(For Basic mode only)  
Select this check box to include the header of the input flow in the output display.

**Print component unique name in front of each output row**  
(For Basic mode only)  
Select this check box to show the unique name the component in front of each output row to differentiate outputs in case several `tLogRow` components are used.

**Print schema column name in front of each value**  
(For Basic mode only)  
Select this check box to retrieve column labels from output schema.

**Use fixed length for values**  
(For Basic mode only)  
Select this check box to set a fixed width for the value display.

### Global Variables

| Global Variables | NB_LINE: the number of rows processed. This is an After variable and it returns an integer.  
|                  | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
|                  | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
|                  | To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
|                  | For further information about variables, see Talend Studio User Guide. |

### Usage

| Usage rule | This component can be used as intermediate step in a data flow or as a end object in the Job flowchart. |

### Related scenarios

For related scenarios, see:

- Reading master data from an MDM hub on page 2149.
- Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
• Catching messages triggered by a tWarn component on page 1982.
• Catching the message triggered by a tDie component on page 1984.
tLoop

Executes a task or a Job automatically, based on a loop

**tLoop Standard properties**

These properties are used to configure tLoop running in the Standard Job framework.

The Standard tLoop component belongs to the Orchestration family.

The component in this framework is generally available.

**Basic settings**

| Loop Type | Select a type of loop to be carried out: either For or While.  
For: The task or Job is carried out for the defined number of iteration  
While: The task or Job is carried out until the condition is met. |
|-----------|------------------------------------------------------------------|
| For       | • From: enter the first instance number which the loop should start from. A start instance number of 2 with a step of 2 means the loop takes on every even number instance.  
• To: enter the last instance number which the loop should finish with.  
• Step: enter the step the loop should be incremented of. A step of 2 means every second instance.  
• Values are increasing: select this check box to only allow an increasing sequence. Deselect this check box to only allow a decreasing sequence. |
| While     | • Declaration: enter an expression initiating the loop.  
• Condition: enter the condition that should be met for the loop to end.  
• Iteration: enter the expression showing the operation to be performed at each loop. |

**Global Variables**

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
CURRENT_VALUE: the current value. Only available for a For type loop. This is a Flow variable and it returns an integer. |
CURRENT_ITERATION: the sequence number of the current iteration. This is a Flow variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tLoop is to be used as a start component and can only be used with an iterate connection to the next component.</th>
</tr>
</thead>
</table>

Connections

Outgoing links (from this component to another):
- **Row**: Iterate.
- **Trigger**: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error.

Incoming links (from one component to this one):
- **Row**: Iterate;
- **Trigger**: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error; Synchronize; Parallelize.

For further information regarding connections, see Talend Studio User Guide.

Scenario: Job execution in a loop

This scenario describes a Job composed of a parent Job and a child Job. The parent Job implements a loop which executes a child Job five times, with a pause of three seconds between each two executions.

Procedure

1. Create a Parent Job, and drop tLoop, tRunJob, and tSleep components to the workspace.
2. Connect tLoop to tRunJob using a Row > Iterate connection, and connect tRunJob to tSleep using a Row > Main connection.
3. Create a Child Job, and drop `tRowGenerator` and `tLogRow` components to the workspace.

4. Connect `tRowGenerator` to `tLogRow` using a Row > Main connection.

5. On the Basic settings view of the `tLoop` component, choose For loop type and type in the instance number to start from (1), to finish with (5) and the step (1).

6. On the Basic settings view of the `tRunJob` component, select Child Job in the Job field.

7. On the Basic settings view of the `tSleep` component, enter 3 in the Pause field.

8. Double-click `tRowGenerator` to open the schema editor. Then click the plus button to add four new columns:
• *id*, to generate sequence numbers
• *firstname*, to generate random first names
• *lastname*, to generate random last names
• *city*, to generate random city names

For more information about **tRowGenerator**, see *tRowGenerator* on page 3281.

9. Press **F6** to run the *Parent Job*.

The *Child Job* will be executed five times with a three-second pause between each two executions, displaying random personal information on the *Run* console as configured in the **tRowGenerator** component.
**tMap**

Transforms and routes data from single or multiple sources to single or multiple destinations.

**Tip:**
There is no order among the output flows of tMap. To make the output flows to be executed one by one, you can output them to temporary files or memory, and then read and insert them into files or databases using different subjobs linked by Trigger > OnSubjobOK connections.

**tMap Standard properties**

These properties are used to configure tMap running in the Standard Job framework.

The Standard tMap component belongs to the Processing family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Map editor</th>
<th>It allows you to define the tMap routing and transformation properties.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If needed, click the button at the top of the input area to open the [Property Settings] dialog box, which provides the following options:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Die on error</strong>: Select this check box if you want to kill the Job if there is an error. This check box is selected by default.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Enable Auto-Conversion of types</strong>: If your input and output columns across a mapping are of different data types, select this check box to enable automatic type conversion at the run time to avoid compiling errors.</td>
</tr>
<tr>
<td></td>
<td>This option is enabled by default if the Enable Auto-Conversion of types check box is selected in the Project Settings view when this component is added. You can also override the default conversion behavior of this component by setting conversion rules in the Project Settings view. For more information, see Talend Studio User Guide.</td>
</tr>
<tr>
<td></td>
<td><strong>Note that auto conversion between Date and BigDecimal is not supported.</strong></td>
</tr>
<tr>
<td></td>
<td>• <strong>Store on disk</strong>: The options provided in this area are identical to the relevant options provided on the Basic settings and Advanced settings tabs respectively. Settings made in the [Property Settings] dialog box are reflected in the respective tab views and vice versa.</td>
</tr>
</tbody>
</table>
### Mapping links display as

| Mapping links display as | Auto: the default setting is curves links  
| Curves: the mapping display as curves  
| Lines: the mapping displays as straight lines. This last option allows to slightly enhance performance. |

### Temp data directory path

| Temp data directory path | Enter the path where you want to store the temporary data generated for lookup loading. For more information on this folder, see *Talend Studio User Guide*. |

### Preview

| Preview | The preview is an instant shot of the Mapper data. It becomes available when Mapper properties have been filled in with data. The preview synchronization takes effect only after saving changes. |

### Advanced settings

| Max buffer size (nb of rows) | Type in the size of physical memory, in number of rows, you want to allocate to processed data. |
| Ignore trailing zeros for BigDecimal | Select this check box to ignore trailing zeros for BigDecimal data. |
| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

### Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
| | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
| | To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
| | For further information about variables, see *Talend Studio User Guide*. |

### Usage

| Usage rule | Possible uses are from a simple reorganization of fields to the most complex Jobs of data multiplexing or demultiplexing transformation, concatenation, inversion, filtering and more... |
| Limitation | The use of **tMap** supposes minimum Java knowledge in order to fully exploit its functionalities. |
Mapping data using a filter and a simple explicit join

The Job described below aims at reading data from a csv file with its schema stored in the Repository, looking up at a reference file, the schema of which is also stored in the Repository, then extracting data from these two files based on a defined filter to an output file and reject files.

Linking the components

Procedure

1. Drop two `tFileInputDelimited` components, `tMap` and three `tFileOutputDelimited` components onto the design workspace.
2. Rename the two `tFileInputDelimited` components as `Cars` and `Owners`, either by double-clicking the label in the design workspace or via the View tab of the Component view.
3. Connect the two input components to `tMap` using Row > Main connections and label the connections as `Cars_data` and `Owners_data` respectively.
4. Connect `tMap` to the three output components using Row > New Output (Main) connections and name the output connections as `Insured`, `Reject_NoInsur` and `Reject_OwnerID` respectively.

Configuring the components

Procedure

1. Double-click the `tFileInputDelimited` component labelled `Cars` to display its Basic settings view.
2. Select Repository from the Property type list and select the component's schema, cars in this scenario, from the [Repository Content] dialog box. The rest fields are automatically filled.

3. Double-click the component labelled Owners and repeat the setting operation. Select the appropriate metadata entry, owners in this scenario.

   **Note:**
   
   In this scenario, the input schemas are stored in the Metadata node of the Repository tree view for easy retrieval. For further information regarding metadata creation in the Repository, see Talend Studio User Guide.

4. Double-click the `tMap` component to open the Map Editor.

   Note that the input area is already filled with the defined input tables and that the top table is the main input table, and the respective row connection labels are displayed on the top bar of the table.

5. Create a join between the two tables on the ID_Owner column by simply dropping the ID_Owner column from the Cars_data table onto the ID_Owner column in the Owners_data table.

6. Define this join as an inner join by clicking the `tMap settings` button, clicking in the Value field for Join Model, clicking the small button that appears in the field, and selecting Inner Join from the [Options] dialog box.
7. Drag all the columns of the *Cars_data* table to the *Insured* table.

8. Drag the *ID_Owner*, *Registration*, and *ID_Reseller* columns of the *Cars_data* table and the *Name* column of the *Owners_data* table to the *Reject_NoInsur* table.

9. Drag all the columns of the *Cars_data* table to the *Reject_OwnerID* table.

   For more information regarding data mapping, see *Talend Studio User Guide*.

10. Click the plus arrow button at the top of the *Insured* table to add a filter row.

    Drag the *ID_Insurance* column of the *Owners_data* table to the filter condition area and enter the formula meaning 'not undefined': `Owners_data.ID_Insurance != null`.

    With this filter, the *Insured* table will gather all the records that include an insurance ID.
11. Click the **tMap settings** button at the top of the *Reject_NoInsur* table and set **Catch output reject** to **true** to define the table as a standard reject output flow to gather the records that do not include an insurance ID.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catch output reject</td>
<td>true</td>
</tr>
<tr>
<td>Catch lookup inner join reject</td>
<td>false</td>
</tr>
<tr>
<td>Schema Type</td>
<td>Built-In</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cars.data.ID_Owner</td>
<td>ID_Owner</td>
</tr>
<tr>
<td>Cars.data.Registration</td>
<td>Registration</td>
</tr>
<tr>
<td>Cars.data.ID_Reseller</td>
<td>ID_Reseller</td>
</tr>
<tr>
<td>Owners.data.Name</td>
<td>Name</td>
</tr>
</tbody>
</table>

12. Click the **tMap settings** button at the top of the *Reject_OwnerID* table and set **Catch lookup inner join reject** to **true** so that this output table will gather the records from the *Cars_data* flow with missing or unmatched owner IDs.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catch output reject</td>
<td>false</td>
</tr>
<tr>
<td>Catch lookup inner join reject</td>
<td>true</td>
</tr>
<tr>
<td>Schema Type</td>
<td>Built-In</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cars.data.ID_Owner</td>
<td>ID_Owner</td>
</tr>
<tr>
<td>Cars.data.Registration</td>
<td>Registration</td>
</tr>
<tr>
<td>Cars.data.Make</td>
<td>Make</td>
</tr>
<tr>
<td>Cars.data.Color</td>
<td>Color</td>
</tr>
<tr>
<td>Cars.data.ID_Reseller</td>
<td>ID_Reseller</td>
</tr>
</tbody>
</table>

Click **OK** to validate the mappings and close the **Map Editor**.

13. Double-click each of the output components, one after the other, to define their properties. If you want a new file to be created, browse to the destination output folder, and type in a file name including the extension.
Select the **Include header** check box to reuse the column labels from the schema as header row in the output file.

**Executing the Job**

**Procedure**

1. Press **Ctrl + S** to save your Job.
2. Press **F6** to run the Job.

The output files are created, which contain the relevant data as defined.

<table>
<thead>
<tr>
<th>Insured_all.csv</th>
<th>Not insured.csv</th>
<th>No_Owner_ID.csv</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_Owner;Registration;ID_Reseller;Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>9;DCI 217;13;William PIECE</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>16;ZYT 702;48;Rutherford HOOVER</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>30;ZST 904;37;Hillard WASHINGTON</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>34;ZII 397;97;Theodore WASHINGTON</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>36;ZST 196;43;Andrew ADAMS</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>44;ZFL 67;59;Franklin MADISON</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>49;DZ 760;52;Benjamin QUINCY</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>52;XMQ 503;25;Martin COOLIDGE</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>66;GNH 801;101;Richard MCNROE</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>70;ZII 771;51;Ronald JEFFERSON</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>72;TBI 459;26;Lyndon EISENHOWER</td>
<td></td>
</tr>
</tbody>
</table>

**Mapping data using inner join rejections**

This scenario, based on scenario 1, adds one input file containing details about resellers and extra fields in the main output table. Two filters on inner joins are added to gather specific rejections.

**Linking the components**

**Procedure**

1. Drop a **tFileInputDelimited** component and a **tFileOutputDelimited** component to the design workspace, and label the components as **Resellers** and **No_Reseller_ID** respectively.
2. Connect it to the Mapper using a **Row > Main** connection, and label the connection as **Resellers_data**.
3. Connect the **tMap** component to the new **tFileOutputDelimited** component by using the **Row** connection named **Reject_ResellerID**.
Configuring the components

Procedure

1. Double-click the Resellers component to display its Basic settings view.

2. Select Repository from the Property type list and select the component’s schema, resellers in this scenario, from the [Repository Content] dialog box. The rest fields are automatically filled.

   **Note:**
   In this scenario, the input schemas are stored in the Metadata node of the Repository tree view for easy retrieval. For further information regarding metadata creation in the Repository, see Talend Studio User Guide.

3. Double-click the tMap component to open the Map Editor.
   Note that the schema of the new input component is already added in the Input area.

4. Create a join between the main input flow and the new input flow by dropping the ID_Reseller column of the Cars_data table to the ID_Reseller column of the Resellers_data table.
5. Click the **tMap settings** button at the top of the *Resellers_data* table and set **Join Model** to **Inner Join**.

6. Drag all the columns except *ID_Reseller* of the *Resellers_data* table to the main output table, *Insured*. 
Note:
When two inner joins are defined, you either need to define two different inner join reject tables to differentiate the two rejections or, if there is only one inner join reject output, both inner join rejections will be stored in the same output.

7. Click the [+] button at the top of the output area to add a new output table, and name this new output table `Reject_ResellerID`.

8. Drag all the columns of the `Cars_data` table to the `Reject_ResellerID` table.

9. Click the `tMap settings` button and select `Catch lookup inner join reject` to `true` to define this new output table as an inner join reject output. If the defined inner join cannot be established, the information about the relevant cars will be gathered through this output flow.

10. Now apply filters on the two Inner Join reject outputs, in order for to distinguish the two types of rejection.
   In the first Inner Join output table, `Reject_OwnerID`, click the plus arrow button to add a filter line and fill it with the following formula to gather only owner ID related rejection: `Owners_data.ID_Owner==null`

11. In the second Inner Join output table, `Reject_ResellerID`, repeat the same operation using the following formula: `Resellers_data.ID_Reseller==null`
Click OK to validate the map settings and close the Mapper Editor.

12. Double-click the No_Reseller_ID component to display its Basic settings view.

Specify the output file path and select the Include Header check box, and leave the other parameters as they are.

13. To demonstrate the work of the Mapper, in this example, remove reseller IDs 5 and 8 from the input file Resellers.csv.

**Executing the Job**

**Procedure**

1. Press Ctrl + S to save your Job.
2. Press F6 to run the Job.

**Results**

The four output files are all created in the specified folder, containing information as defined. The output file No_Reseller_ID.csv contains the cars information related to reseller IDs 5 and 8, which are missing in the input file Resellers.csv.
As third advanced use scenario, based on the scenario 2, add a new Input table containing Insurance
details for example.

Set up an Inner Join between two lookup input tables (Owners and Insurance) in the Mapper to create
a cascade lookup and hence retrieve Insurance details via the Owners table data.

Advanced mapping using filters, explicit joins and rejections

This scenario introduces a Job that allows you to find BMW owners who have two to six children
(inclusive), for sales promotion purpose for example.

Linking the components

Procedure

1. Drop three tFileInputDelimited components, a tMap component, and two tFileOutputDelimited
   components from the Palette onto the design workspace, and label them to best describe their
   functions.

2. Connect the input components to the tMap using Row > Main connections.
   Pay attention to the file you connect first as it will automatically be set as Main flow, and all the
   other connections will be Lookup flows. In this example, the connection for the input component
   Owners is the Main flow.
Configuring the components

Procedure

1. Define the properties of each input components in the respective Basic settings view. Define the properties of Owners.

   ![Owners Component](image)

   Property Type: Repository, DELIMlosures
   "When the input source is a stream or a zip file, footer and random shouldn't be bigger than 0."
   Row Separator: "\n"
   Field Separator: ";"
   Schema: Repository, DELIMlosures - metadata
   No Skip empty rows, Uncompress as zip file, Die on error

2. Select Repository from the Property type list and select the component's schema, owners in this scenario, from the [Repository Content] dialog box. The rest fields are automatically filled.

   Note:
   In this scenario, the input schemas are stored in the Metadata node of the Repository tree view for easy retrieval. For further information regarding metadata creation in the Repository, see Talend Studio User Guide.

   In the same way, set the properties of the other input components: Cars and Resellers. These two Lookup flows will fill in secondary (lookup) tables in the input area of the Map Editor.

3. Then double-click the tMap component to launch the Map Editor and define the mappings and filters.

   Set an explicit join between the Main flow Owner and the Lookup flow Cars by dropping the ID_Owner column of the Owners table to the ID_Owner column of the Cars table.
The explicit join is displayed along with a hash key.

4. In the **Expr. Key** field of the **Make** column, type in a filter. In this use case, simply type in "BMW" as the search is focused on the owners of this particular make.

5. Implement a cascading join between the two lookup tables **Cars** and **Resellers** on the **ID_Reseller** column in order to retrieve resellers' information.

6. As you want to reject the null values into a separate table and exclude them from the standard output, click the **tMap settings** button and set **Join Model** to **Inner Join** in each of the **Lookup** tables.
7. In the tMap settings, you can set **Match Model** to **Unique match**, **First match**, or **All matches**. In this use case, the **All matches** option is selected. Thus if several matches are found in the Inner Join, rows matching the explicit join as well as the filter, all of them will be added to the output flow (either in rejection or the regular output).

   **Note:**

   The **Unique match** option functions as a **Last match**. The **First match** and **All matches** options function as named.

8. On the output area of the **Map Editor**, click the plus button to add two tables, one for the full matches and the other for the rejections.

9. Drag all the columns of the **Owners** table, the **Registration**, **Make** and **Color** columns of the **Cars** table, and the **ID_Reseller** and **Name_Reseller** columns of the **Resellers** table to the main output table.

10. Drag all the columns of the **Owners** table to the reject output table.

11. Click the **Filter** button at the top of the main output table to display the **Filter** expression area.

   Type in a filter statement to narrow down the number of rows loaded in the main output flow. In this use case, the statement reads: `Owners.Children_Nr >= 2 && Owners.Children_Nr <= 6`.

12. In the reject output table, click the **tMap settings** button and set the reject types.

   Set **Catch output reject** to **true** to collect data about BMW car owners who have less than two or more than six children.
Set **Catch lookup inner join reject** to **true** to collect data about owners of other car makes and owners for whom the reseller information is not found.

Click **OK** to validate the mappings and close the **Map Editor**.

On the design workspace, right-click the **tMap** and pull the respective output link to the relevant output components.

13. Define the properties of the output components in their respective **Basic settings** view.

In this use case, simply specify the output file paths and select the **Include Header** check box, and leave the other parameters as they are.
Executing the Job

Procedure

1. Press **Ctrl + S** to save your Job.
2. Press **F6** to run it.

The main output file contains the information related to BMW owners who have two to six children, and the reject output file contains the information about the rest of the car owners.

```
ID_Owner;Name;ID_Insurance;ChildrenNr;Registration;Make;Color;
ID_Reseller;Name_Reseller
15;Benjamin ROOSEVELT;103;2;FJ3 72;BMW;yellow;80;Lyndon TAL
29;Bill GRANT;70;2;TJS 530;BMW;grey;94;Grover ROOSEVELT
426;Chester VAN BUREN;184;2;PIZ 361;BMW;yellow;6;Benjamin HARDING
38;Andrew ADAMS;16;3;G7T 196;BMW;yellow;43;William WASHINGTON
47;Bill ARTHUR;145;6;REZ 45;BMW;white;43;William WASHINGTON
70;Ronald JEFFERSON;70;3;Z7I 771;BMW;green;51;Abraham JACKSON
```

Advanced mapping with filters and different rejections

This scenario is a modified version of the preceding scenario. It describes a Job that applies filters to limit the search to BMW and Mercedes owners who have two to six children and divides unmatched data into different reject output flows.

Linking the components

Procedure

1. Take the same Job as in Advanced mapping using filters, explicit joins and rejections on page 2006.
2. Drop a new **tFileOutputDelimited** component from the **Palette** on the design workspace, and name it **Rejects_BMW_Mercedes** to present its functionality.
3. Connect the **tMap** component to the new output component using a **Row** connection and label the connection according to the functionality of the output component. This connection label will appear as the name of the new output table in the **Map Editor**.
4. Relabel the existing output connections and output components to reflect their functionality.

The existing output tables in the **Map Editor** will be automatically renamed according to the connection labels. In this example, relabel the existing output connections **BMW_Mercedes_withChildren** and **Owners_Other_Makes** respectively.
Configuring the components

Procedure

1. Double-click the tMap component to launch the Map Editor to change the mappings and the filters. Note that the output area contains a new, empty output table named Rejects_BMW_Mercedes. You can adjust the position of the table by selecting it and clicking the Up or Down arrow button at the top of the output area.

2. Remove the Expr. key filter ("BMW") from the Cars table in the input area.

3. Click the Filters button to display the Filter field, and type in a new filter to limit the search to BMW or Mercedes car makes. The statement reads as follows: Cars.Make.equals("BMW") || Cars.Make.equals("Mercedes")
4. Select all the columns of the main output table and drop them down to the new output table. Alternatively, you can also drag the corresponding columns from the relevant input tables to the new output table.

5. Click the tMap settings button at the top of the new output table and set Catch output reject to true to collect data about BMW and Mercedes owners who have less than two or more than six children.

6. In the Owners_Other_Makes table, set Catch lookup inner join reject to true to collect data about owners of other car makes and owners for whom the reseller information is not found.

7. Click OK to validate the mappings and close the Map Editor.

8. Define the properties of the output components in their respective Basic settings view.
In this use case, simply specify the output file paths and select the **Include Header** check box, and leave the other parameters as they are.

### Executing the Job

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to run it.

The output files contain content of the main output flow shows that the filtered rows have correctly been passed on.

### Advanced mapping with lookup reload at each row

The following scenario describes a Job that retrieves people details from a lookup database, based on a join on the age. The main flow source data is read from a MySQL database table called `people_age` that contains people details such as numeric id, alphanumeric first name and last name and numeric age. The people age is either 40 or 60. The number of records in this table is intentionally restricted.

The reference or lookup information is also stored in a MySQL database table called `large_data_volume`. This lookup table contains a number of records including the city where people from the main flow have been to. For the sake of clarity, the number of records is restricted but, in a normal use, the usefulness of the feature described in the example below is more obvious for very large reference data volume.
To optimize performance, a database connection component is used in the beginning of the Job to open the connection to the lookup database table in order not to do that every time we want to load a row from the lookup table.

An Expression Filter is applied to this lookup source flow, in order to select only data from people whose age is equal to 60 or 40. This way only the relevant rows from the lookup database table are loaded for each row from the main flow.

Therefore this Job shows how, from a limited number of main flow rows, the lookup join can be optimized to load only results matching the expression key.

Note:
Generally speaking, as the lookup loading is performed for each main flow row, this option is mainly interesting when a limited number of rows is processed in the main flow while a large number of reference rows are to be looked up to.

The join is solved on the age field. Then, using the relevant loading option in the tMap component editor, the lookup database information is loaded for each main flow incoming row.

For this Job, the metadata has been prepared for the source and connection components. For more information on how to set up the DB connection schema metadata, see the relevant section in the Talend Studio User Guide.

This Job is formed with five components, four database components and a mapping component.

**Linking the components**

**Procedure**

1. Drop the DB Connection under the Metadata node of the Repository to the design workspace. In this example, the source table is called people_age.
2. Select tMysqlInput from the list that pops up when dropping the component.
3. Drop the lookup DB connection table from the Metadata node to the design workspace selecting tMysqlInput from the list that pops up. In this Job, the lookup is called large_data_volume.
4. The same way, drop the DB connection from the Metadata node to the design workspace selecting tMysqlConnection from the list that pops up. This component creates a permanent connection to the lookup database table in order not to do that every time we want to load a row from the lookup table.

5. Then pick the tMap component from the Processing family, and the tMysqlOutput and tMysqlCommit components from the Database family in the Palette to the right hand side of the editor.

6. Now connect all the components together. To do so, right-click the tMysqlInput component corresponding to the people table and drag the link towards tMap.

7. Release the link over the tMap component, the main row flow is automatically set up.

8. Rename the Main row link to people, to identify more easily the main flow data.

9. Perform the same operation to connect the lookup table (large_data_volume) to the tMap component and the tMap to the tMysqlOutput component.

10. A dialog box prompts for a name to the output link. In this example, the output flow is named: people_mixandmatch.

11. Rename also the lookup row connection link to large_volume, to help identify the reference data flow.

12. Connect tMysqlConnection to tMysqlInput using the trigger link OnSubjobOk.

13. Connect the tMysqlInput component to the tMysqlCommit component using the trigger link OnSubjobOk.

Configuring the components

Procedure

1. Double-click the tMap component to open the graphical mapping editor.
2. The Output table (that was created automatically when you linked the \texttt{tMap} to the \texttt{tMySQLOutput}) will be formed by the matching rows from the lookup flow (\textit{large_data_volume}) and the main flow (\textit{people_age}).

Select the main flow rows that are to be passed on to the output and drag them over to paste them in the Output table (to the right hand side of the mapping editor).

In this example, the selection from the main flow include the following fields: \textit{id}, \textit{first_name}, \textit{last_Name} and \textit{age}.

From the lookup table, the following column is selected: \textit{city}.

Drop the selected columns from the input tables (\textit{people} and \textit{large_volume}) to the output table.

3. Now set up the join between the main and lookup flows.

Select the \textit{age} column of the main flow table (on top) and drag it towards the \textit{age} column of the lookup flow table (\textit{large_volume} in this example).

A key icon appears next to the linked expression on the lookup table. The join is now established.

4. Click the \texttt{tMap settings} button, click the three-dot button corresponding to \textbf{Lookup Model}, and select the \textit{Reload at each row} option from the \texttt{[Options]} dialog box in order to reload the lookup for each row being processed.
5. In the same way, set **Match Model** to **All matches** in the Lookup table, in order to gather all instances of *age* matches in the output flow.

6. Now implement the filtering, based on the *age* column, in the Lookup table. The **GlobalMapKey** field is automatically created when you selected the **Reload at each row** option. Indeed you can use this expression to dynamically filter the reference data in order to load only the relevant information when joining with the main flow.

As mentioned in the introduction of the scenario, the main flow data contains only people whose age is either 40 or 60. To avoid the pain of loading all lookup rows, including ages that are different from 40 and 60, you can use the main flow age as global variable to feed the lookup filtering.
7. Drop the Age column from the main flow table to the Expr. field of the lookup table.

8. Then in the globalMap Key field, put in the variable name, using the expression. In this example, it reads: "people.Age"

   Click OK to save the mapping setting and go back to the design workspace.

9. To finalize the implementation of the dynamic filtering of the lookup flow, you need now to add a WHERE clause in the query of the database input.

10. At the end of the Query field, following the Select statement, type in the following WHERE clause: WHERE AGE = ""+(Integer)globalMap.get("people.Age")+""
11. Make sure that the type corresponds to the column used as variable. In this use case, Age is of Integer type. And use the variable the way you set in the globalMap key field of the map editor.

12. Double-click the tMysqloutput component to define its properties.

13. Select the Use an existing connection check box to leverage the created DB connection. Define the target table name and relevant DB actions.

**Executing the Job**

**Procedure**

1. Press Ctrl + S to save the Job.
2. Click the Run tab at the bottom of the design workspace, to display the Job execution tab.
3. From the Debug Run view, click the Traces Debug button to view the data processing progress.
   For more comfort, you can maximize the Job design view while executing by simply double-clicking on the Job name tab.

The lookup data is reloaded for each of the main flow's rows, corresponding to the age constraint. All age matches are retrieved in the lookup rows and grouped together in the output flow. Therefore if you check out the data contained in the newly created people_mixandmatch table, you will find all the age duplicates corresponding to different individuals whose age equals to 60 or 40 and the city where they have been to.
Mapping with join output tables

The following scenario describes a Job that processes reject flows without separating them from the main flow.

Linking the components

Procedure

1. In the Repository tree view, click Metadata > File delimited. Drag and drop the customers metadata onto the workspace.

   The customers metadata contains information about customers, such as their ID, their name or their address, etc.

   For more information about centralizing metadata, see Talend Studio User Guide.

2. In the dialog box that asks you to choose which component type you want to use, select tFileInputDelimited and click OK.

3. Drop the states metadata onto the design workspace. Select the same component in the dialog box and click OK.

   The states metadata contains the ID of the state, and its name.

4. Drop a tMap and two tLogRow components from the Palette onto the design workspace.

5. Connect the customers component to the tMap, using a Row > Main connection.

6. Connect the states component to the tMap, using a Row > Main connection. This flow will automatically be defined as Lookup.
Configuring the components

Procedure

1. Double-click the tMap component to open the Map Editor.

   Drop the idState column from the main input table to the idState column of the lookup table to create a join.

   Click the tMap settings button and set Join Model to Inner Join.

2. Click the Property Settings button at the top of the input area to open the [Property Settings] dialog box, and clear the Die on error check box in order to handle the execution errors.

   The ErrorReject table is automatically created.
3. Select the id, idState, RegTime and RegisterTime in the input table and drag them to the ErrorReject table.

4. Click the [+] button at the top right of the editor to add an output table. In the dialog box that opens, select New output. In the field next to it, type in the name of the table, out1. Click OK.

5. Drag the following columns from the input tables to the out1 table: id, CustomerName, idState, and LabelState.

Add two columns, RegTime and RegisterTime, to the end of the out1 table and set their date formats: "dd/MM/yyyy HH:mm" and "yyyy-MM-dd HH:mm:ss.SSS" respectively.

6. Click in the Expression field for the RegTime column, and press Ctrl+Space to display the auto-completion list. Find and double-click TalendDate.parseDate. Change the pattern to ("dd/MM/yyyy HH:mm", row1.RegTime).

7. Do the same thing for the RegisterTime column, but change the pattern to ("yyyy-MM-dd HH:mm:ss.SSS", row1.RegisterTime).

8. Click the [+] button at the top of the output area to add an output table. In the dialog box that opens, select Create join table from, choose Out1, and name it rejectInner. Click OK.

9. Click the tMap settings button and set Catch lookup inner join reject to true in order to handle rejects.
10. Drag the id, CustomerName, and idState columns from the input tables to the corresponding columns of the rejectInner table.

   Click in the Expression field for the LabelState column, and type in "UNKNOWN".

11. Click in the Expression field for the RegTime column, press Ctrl+Space, and select TalendDate.parseDate. Change the pattern to ("dd/MM/yyyy HH:mm", row1.RegTime).

12. Click in the Expression field for the RegisterTime column, press Ctrl+Space, and select TalendDate.parseDate, but change the pattern to ("yyyy-MM-dd HH:mm:ss.SSS", row1.RegisterTime).

   If the data from row1 has a wrong pattern, it will be returned by the ErrorReject flow.

   ![Out1](image1)

   ![Join Table](image2)

   Click OK to validate the changes and close the editor.

13. Double-click the first tLogRow component to display its Component view.

   Click Sync columns to retrieve the schema structure from the mapper if needed.

   In the Mode area, select Table.

   Do the same thing with the second tLogRow.

### Executing the Job

**Procedure**

1. Press Ctrl + S to save your Job.
2. Press F6 to execute it.
Results

The **Run** console displays the main out flow and the ErrorReject flow. The main output flow unites both valid data and inner join rejects, while the ErrorReject flow contains the error information about rows with unparsable date formats.
tMapRDBClose

Closes an MapRDB connection you have established in a same Job.

**tMapRDBClose Standard properties**

These properties are used to configure tMapRDBClose running in the Standard Job framework.

The Standard tMapRDBClose component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Component list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the <strong>tMapRDBConnection</strong> component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>tStat Catcher Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](https://www.talend.com). |

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component is to be used along with MapRDB components, especially with <strong>tMapRDBConnection</strong>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prerequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before starting, ensure that you have met the Loopback IP prerequisites expected by your database. The Hadoop distribution must be properly installed, so as to guarantee the interaction with <a href="https://www.talend.com">Talend Studio</a>. The following list presents MapR related information for example.</td>
</tr>
</tbody>
</table>
• Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

• Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

Related scenario

This component is similar to tHBaseClose. For a scenario about how to use tHBaseClose, see Scenario: Exchanging customer data with HBase on page 1424.
tMapRDBConnection

Establishes a MapRDB connection to be reused by other MapRDB components in a same Job.

**tMapRDBConnection Standard properties**

These properties are used to configure tMapRDBConnection running in the Standard Job framework. The Standard tMapRDBConnection component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored.</td>
</tr>
<tr>
<td></td>
<td>The properties are stored centrally under the Hadoop Cluster node of the Repository tree.</td>
</tr>
</tbody>
</table>

**Distribution and Version**

Select the MapR distribution to be used. Only MapR V5.2 onwards is supported by the MapRDB components.

If the distribution you need to use with your MapRDB database is not officially supported by this MapRBD component, that is to say, this distribution is MapR but is not listed in the Version drop-down list of this components or this distribution is not MapR at all, select Custom.

1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the Import from existing version option to take an existing distribution as base to add the jars required by your distribution.
Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see Connecting to a custom Hadoop distribution on page 1589.

<table>
<thead>
<tr>
<th>Hadoop version of the distribution</th>
<th>This list is displayed only when you have selected Custom from the distribution list to connect to a cluster not yet officially supported by the Studio. In this situation, you need to select the Hadoop version of this custom cluster, that is to say, Hadoop 1 or Hadoop 2.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zookeeper quorum</td>
<td>Type in the name or the URL of the Zookeeper service you use to coordinate the transaction between your Studio and your database. Note that when you configure the Zookeeper, you might need to explicitly set the zookeeper.znode.parent property to define the path to the root znode that contains all the znodes created and used by your database; then select the Set Zookeeper znode parent check box to define this property.</td>
</tr>
<tr>
<td>Zookeeper client port</td>
<td>Type in the number of the client listening port of the Zookeeper service you are using.</td>
</tr>
<tr>
<td>Inspect the classpath for configurations</td>
<td>Select this check box to allow the component to check the configuration files in the directory you have set with the $HADOOP_CONF_DIR variable and directly read parameters from these files in this directory. This feature allows you to easily change the Hadoop configuration for the component to switch between different environments, for example, from a test environment to a production environment. In this situation, the fields or options used to configure Hadoop connection and/or Kerberos security are hidden. If you want to use certain parameters such as the Kerberos parameters but these parameters are not included in these Hadoop configuration files, you need to create a file called talend-site.xml and put this file into the same directory defined with $HADOOP_CONF_DIR. This talend-site.xml file should read as follows:</td>
</tr>
</tbody>
</table>
<!-- Put site-specific property overrides in this file. -->
<configuration>
    <property>
        <name>talend.kerberos.authentication</name>
        <value>kinit</value>
        <description> Set the Kerberos authentication method to use. Valid values are: kinit or keytab. </description>
    </property>
    <property>
        <name>talend.kerberos.keytab.principal</name>
        <value>user@BIGDATA.COM</value>
        <description> Set the keytab's principal name. </description>
    </property>
    <property>
        <name>talend.kerberos.keytab.path</name>
        <value>/kdc/user.keytab</value>
        <description> Set the keytab's path. </description>
    </property>
    <property>
        <name>talend.encryption</name>
        <value>none</value>
        <description> Set the encryption method to use. Valid values are: none or ssl. </description>
    </property>
    <property>
        <name>talend.ssl.trustStore.path</name>
        <value>ssl</value>
        <description> Set SSL trust store path. </description>
    </property>
    <property>
        <name>talend.ssl.trustStore.password</name>
        <value>ssl</value>
        <description> Set SSL trust store password. </description>
    </property>
</configuration>

The parameters read from these configuration files override the default ones used by the Studio. When a parameter does not exist in these configuration files, the default one is used.
Use kerberos authentication

If the database to be used is running with Kerberos security, select this check box, then, enter the principal names in the displayed fields. You should be able to find the information in the hbase-site.xml file of the cluster to be used.

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the Force MapR ticket authentication check box and the Use Kerberos authentication check box clear, and then MapR should be able to automatically find that ticket on the fly.

If you need to use a Kerberos keytab file to log in, select **Use a keytab to authenticate**. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the Principal field and the access path to the keytab file itself in the Keytab field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is user1 and the principal to be used is guest; in this situation, ensure that user1 has the right to read the keytab file to be used.

For further information about how Kerberos can be configured for your database in a MapR cluster, see Configuring Kerberos Authentication.

Advanced settings

### Properties

If you need to use custom configuration for your database, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those corresponding ones defined earlier for your database.

For example, you need to define the value of the dfs.replication property as 1 for the database configuration. Then you need to add one row to this table using the plus button and type in the name and the value of this property in this row.

### tStatCatcher Statistics

Select this check box to collect the log data at a component level.
**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see [Talend Studio User Guide](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr). |

**Usage**

| Usage rule | This component is generally used with other MapRDB components, particularly **tMapRDBCReopen**. |

**Prerequisites**

| Prerequisites | Before starting, ensure that you have met the Loopback IP prerequisites expected by your database. The Hadoop distribution must be properly installed, so as to guarantee the interaction with [Talend Studio](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr). The following list presents MapR related information for example. |

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the **Preferences** dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the **Data viewer** to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using. |
Related scenario

This component is similar to **tHBaseConnection**. For a scenario about how to use **tHBaseConnection**, see [Scenario: Exchanging customer data with HBase](#) on page 1424.
tMapRDBInput

Reads data from a given MapRDB database and extracts columns of selection.

tMapRDBInput extracts columns corresponding to schema definition. Then it passes these columns to the next component via a Main row link.

**tMapRDBInput Standard properties**

These properties are used to configure tMapRDBInput running in the Standard Job framework.

The Standard tMapRDBInput component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored.</td>
</tr>
<tr>
<td></td>
<td>The properties are stored centrally under the Hadoop Cluster node of the Repository tree.</td>
</tr>
</tbody>
</table>

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. |

| Distribution and Version | Select the MapR distribution to be used. Only MapR V5.2 onwards is supported by the MapRDB components. |
|                         | If the distribution you need to use with your MapRDB database is not officially supported by this MapRBD component, that is to say, this distribution is MapR but is not listed in the Version drop-down list of this components or this distribution is not MapR at all, select Custom. |
|                         | 1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide. |
|                         | 2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries. |
|                         | In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different |
Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

### Hadoop version of the distribution

This list is displayed only when you have selected **Custom** from the distribution list to connect to a cluster not yet officially supported by the Studio. In this situation, you need to select the Hadoop version of this custom cluster, that is to say, **Hadoop 1** or **Hadoop 2**.

### Zookeeper quorum

Type in the name or the URL of the Zookeeper service you use to coordinate the transaction between your Studio and your database. Note that when you configure the Zookeeper, you might need to explicitly set the `zookeeper.znode.parent` property to define the path to the root znode that contains all the znodes created and used by your database; then select the **Set Zookeeper znode parent** check box to define this property.

### Zookeeper client port

Type in the number of the client listening port of the Zookeeper service you are using.

### Use kerberos authentication

If the database to be used is running with Kerberos security, select this check box, then, enter the principal names in the displayed fields. You should be able to find the information in the `hbase-site.xml` file of the cluster to be used.

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in **Connecting to a security-enabled MapR** on page 1652.
Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

If you need to use a Kerberos keytab file to log in, select **Use a keytab to authenticate**. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is **user1** and the principal to be used is **guest**; in this situation, ensure that **user1** has the right to read the keytab file to be used.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

### Set table Namespace mappings

Enter the string to be used to construct the mapping between an Apache HBase table and a MapR table.
Table name

Type in the name of the table from which you need to extract columns.

Define a row selection

Select this check box and then in the **Start row** and the **End row** fields, enter the corresponding row keys to specify the range of the rows you want the current component to extract.

Different from the filters you can set using **Is by filter** requiring the loading of all records before filtering the ones to be used, this feature allows you to directly select only the rows to be used.

Mapping

Complete this table to map the columns of the table to be used with the schema columns you have defined for the data flow to be processed.

---

### Advanced settings

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

**Properties**

If you need to use custom configuration for your database, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override the corresponding ones used by the Studio.

For example, you need to define the value of the *dfs.replication* property as 1 for the database configuration. Then you need to add one row to this table using the plus button and type in the name and the value of this property in this row.

**Note:**

This table is not available when you are using an existing connection by selecting the **Using an existing connection** check box in the **Basic settings** view.

**Is by filter**

Select this check box to use filters to perform fine-grained data selection from your database, such as selection of keys, or values, based on regular expressions.

Once selecting it, the **Filter** table that is used to define filtering conditions becomes available.

This feature leverages filters provided by HBase and subject to constraints explained in Apache HBase documentation. Therefore, advanced knowledge of HBase is required to make full use of these filters.

**Logical operation**

Select the operator you need to use to define the logical relation between filters. This available operators are:
• **And**: every defined filtering conditions must be satisfied. It represents the relationship `FilterList.Operator.MUST_PASS_ALL`

• **Or**: at least one of the defined filtering conditions must be satisfied. It represents the relationship: `FilterList.Operator.MUST_PASS_ONE`

### Filter

Click the button under this table to add as many rows as required, each row representing a filter. The parameters you may need to set for a filter are:

- **Filter type**: the drop-down list presents pre-existing filter types that are already defined by HBase. Select the type of the filter you need to use.

- **Filter column**: enter the column qualifier on which you need to apply the active filter. This parameter becomes mandatory depending on the type of the filter and of the comparator you are using. For example, it is not used by the **Row Filter** type but is required by the **Single Column Value Filter** type.

- **Filter family**: enter the column family on which you need to apply the active filter. This parameter becomes mandatory depending on the type of the filter and of the comparator you are using. For example, it is not used by the **Row Filter** type but is required by the **Single Column Value Filter** type.

- **Filter operation**: select from the drop-down list the operation to be used for the active filter.

- **Filter Value**: enter the value on which you want to use the operator selected from the **Filter operation** drop-down list.

- **Filter comparator type**: select the type of the comparator to be combined with the filter you are using.

Depending on the **Filter type** you are using, some or each of the parameters become mandatory. For further information, see [HBase filters](#) on page 1417

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
For further information about variables, see Talend Studio User Guide.

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is a start component of a Job and always needs an output link.</th>
</tr>
</thead>
</table>

**Prerequisites**

Before starting, ensure that you have met the Loopback IP prerequisites expected by your database.

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

**Related scenario**

This component is similar to [tHBaseInput](https://www.talend.com/docs/devguide/). For related scenario to tHBaseInput, see Scenario: Exchanging customer data with HBase on page 1424.
tMapRDBOutput

Writes columns of data into a given MapRDB database.

tMapRDBOutput receives data from its preceding component, creates a table in a given MapRDB database and writes the received data into this table.

**tMapRDBOutput Standard properties**

These properties are used to configure tMapRDBOutput running in the Standard Job framework. The Standard tMapRDBOutput component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-In: No property data stored centrally. Repository: Select the repository file where the properties are stored.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Distribution and Version</th>
<th>Select the MapR distribution to be used. Only MapR V5.2 onwards is supported by the MapRDB components.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the distribution you need to use with your MapRDB database is not officially supported by this MapRBD component, that is to say, this distribution is MapR but is not listed in the Version drop-down list of this components or this distribution is not MapR at all, select Custom.</td>
</tr>
</tbody>
</table>

1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

   In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended
to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th>Hadoop version of the distribution</th>
<th>This list is displayed only when you have selected <strong>Custom</strong> from the distribution list to connect to a cluster not yet officially supported by the Studio. In this situation, you need to select the Hadoop version of this custom cluster, that is to say, <strong>Hadoop 1</strong> or <strong>Hadoop 2</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zookeeper quorum</td>
<td>Type in the name or the URL of the Zookeeper service you use to coordinate the transaction between your Studio and your database. Note that when you configure the Zookeeper, you might need to explicitly set the <code>zookeeper.znode.parent</code> property to define the path to the root znode that contains all the znodes created and used by your database; then select the <strong>Set Zookeeper znode parent</strong> check box to define this property.</td>
</tr>
<tr>
<td>Zookeeper client port</td>
<td>Type in the number of the client listening port of the Zookeeper service you are using.</td>
</tr>
</tbody>
</table>
| Use kerberos authentication | If the database to be used is running with Kerberos security, select this check box, then, enter the principal names in the displayed fields. You should be able to find the information in the `hbase-site.xml` file of the cluster to be used.  

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in **Connecting to a security-enabled MapR** on page 1652.  

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an
existing ticket issued for the same username, leave both the Force MapR ticket authentication check box and the Use Kerberos authentication check box clear, and then MapR should be able to automatically find that ticket on the fly.

If you need to use a Kerberos keytab file to log in, select Use a keytab to authenticate. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the Principal field and the access path to the keytab file itself in the Keytab field. This keytab file must be stored on the machine in which your Job actually runs, for example, on a Talend Jobserver. Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is user1 and the principal name you are using to execute a Job is guest. In this situation, ensure that user1 has the right to read the keytab file to be used.

Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the Repository Content window.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).
Set table Namespace mappings

Enter the string to be used to construct the mapping between an Apache HBase table and a MapR table.

For the valid syntax you can use, see [http://doc.mapr.com/display/MapR40x/Mapping+Table+Namespace+Between+Apache+HBase+Tables+and+MapR+Tables](http://doc.mapr.com/display/MapR40x/Mapping+Table+Namespace+Between+Apache+HBase+Tables+and+MapR+Tables).

<table>
<thead>
<tr>
<th>Table name</th>
<th>Type in the name of the HBase table you need to create.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action on table</td>
<td>Select the action you need to take for creating a table.</td>
</tr>
<tr>
<td>Custom Row Key</td>
<td>Select this check box to use the customized row keys. Once selected, the corresponding field appears. Then type in the user-defined row key to index the rows of the table being created. For example, you can type in &quot;France&quot; + Numeric.sequence(&quot;s1&quot;, 1, 1) to produce the row key series: France1, France2, France3 and so on.</td>
</tr>
<tr>
<td>Families</td>
<td>Complete this table to specify the column or columns to be created and the corresponding column family or families they belong to respectively. The Column column of this table is automatically filled once you have defined the schema.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Use batch mode</th>
<th>Select this check box to activate the batch mode for data processing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>Specify the number of records to be processed in each batch. This field appears only when the Use batch mode check box is selected.</td>
</tr>
<tr>
<td>Properties</td>
<td>If you need to use custom configuration for your database, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override the corresponding ones used by the Studio. For example, you need to define the value of the dfs.replication property as 1 for the database configuration. Then you need to add one row to this table using the plus button and type in the name and the value of this property in this row.</td>
</tr>
</tbody>
</table>

**Note:** This table is not available when you are using an existing connection by selecting the Using an existing connection check box in the Basic settings view.
<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family parameters</td>
<td>Type in the names and, when needs be, the custom performance options of the column families to be created. This feature leverages attributes defined by the HBase data model, so for further explanation about these options, see Apache HBase documentation.</td>
</tr>
<tr>
<td></td>
<td>Note: The parameter Compression type allows you to select the format for output data compression.</td>
</tr>
<tr>
<td>Global Variables</td>
<td></td>
</tr>
<tr>
<td>Global Variables</td>
<td><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td>Usage</td>
<td></td>
</tr>
<tr>
<td>Usage rule</td>
<td>This component is normally an end component of a Job and always needs an input link.</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>Before starting, ensure that you have met the Loopback IP prerequisites expected by your database.</td>
</tr>
<tr>
<td></td>
<td>The Hadoop distribution must be properly installed, so as to guarantee the interaction with <em>Talend Studio</em>. The following list presents MapR related information for example.</td>
</tr>
<tr>
<td></td>
<td>• Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under <em>MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native</em>. For example, the library for Windows is <em>lib\native\MapRClient.dll</em> in the MapR client jar file. For further information, see the following link from MapR: <a href="http://www.mapr.com/">http://www.mapr.com/</a>.</td>
</tr>
</tbody>
</table>
Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

Related scenario

This component is similar to tHBaseOutput. For related scenario to tHBaseOutput, see Scenario: Exchanging customer data with HBase on page 1424.
tMapROjaiOutput

Inserts, replaces or deletes documents in a MapR-DB database to be used as document database, based on the incoming flow from the preceding component in the Job.

You must have properly installed and configured your MapR Client before using this component. A \opt\mapr\conf\mapr-clusters.conf file is automatically generated out of the Client configuration process to store the connection information to your MapR cluster and your MapR-DB database. At runtime, tMapROjaiOutput reads this connection information by itself to connect to the MapR-DB database to be used.

For further information about how to install and configure your MapR client, see Setting up the Client from the MapR documentation.

**tMapROjaiOutput Standard properties**

These properties are used to configure tMapROjaiOutput running in the Standard Job framework.

The Standard tMapROjaiOutput component belongs to the Databases family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file where the properties are stored.</td>
</tr>
<tr>
<td>Table</td>
<td>Enter the name of the table to be processed.</td>
</tr>
<tr>
<td>DB versions</td>
<td>Select the version of your database.</td>
</tr>
</tbody>
</table>

**Use kerberos authentication**

If you are accessing a MapR-DB as-OJAI database running with Kerberos security, select this check box, then, enter the Kerberos principal name and password in the displayed fields.

Every time when you launch your Job, the component submits this authentication information to Kerberos for a new kinit ticket.

- If the MapR ticket security mechanism is also set up along with Kerberos, this component issues a `mapr login` command to obtain a MapR ticket at the same time.
- If only the MapR ticket security mechanism is set up for your cluster while Kerberos is not, keep this check box clear. The component obtains a MapR ticket itself via your MapR Client.

**Use a keytab to authenticate**

Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals.
and encrypted keys. You need to enter the principal to be used in the Principal field and the access path to the keytab file itself in the Keytab field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is *user1* and the principal to be used is *guest*; in this situation, ensure that *user1* has the right to read the keytab file to be used.

---

### Action on table

Select an operation to be performed on the table defined.

- **None**: No operation is carried out.
- **Drop and create table**: The table is removed and created again.
- **Create table**: The table does not exist and gets created.
- **Create table if does not exist**: The table is created if it does not exist.
- **Drop table if exist and create**: The table is removed if it already exists and created again.
- **Truncate**: The table content is deleted.

### Action on data

Select an action to be performed on data of the table defined.

- **Insert**: Add new entries to the table. If duplicates are found, job stops.
- **Replace**: If the table already contains data, delete all the existing data and insert the new data. If the table is empty, insert the new data.
- **Insert or Replace**: It looks at the document IDs, replaces the documents whose IDs exist in both the database and the data to be written, and inserts the documents whose IDs do not exist in the database.
- **Update or insert**: Update the record with the given reference. If the record does not exist in the index pool, a new record would be inserted.
- **Delete**: Remove entries corresponding to the input flow.

### Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

<table>
<thead>
<tr>
<th>Built-In: You create and store the schema locally for this component only. Related topic: see <a href="https://help.talend.com">Talend Studio User Guide</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <a href="https://help.talend.com">Talend Studio User Guide</a>. When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
<tr>
<td><strong>Bulk write</strong> Select this check box to insert, update or remove data in bulk. In the <strong>Bulk write size</strong> field, enter the size of each query group to be processed by MapR-DB.</td>
</tr>
</tbody>
</table>
| **Mapping** Each column of the schema defined for this component represents a field of the documents to be read. In this table, you need to specify the parent nodes of these fields, if any. For example, in the document reading as follows

```json
{
   _id: ObjectId("5099803df3f4948bd2f98391"),
   person: { first: "Joe", last: "Walker" }
}
```

The **first** and the **last** fields have **person** as their parent node but the **_id** field does not have any parent node. So once completed, this Mapping table should read as follows:

<table>
<thead>
<tr>
<th>Column</th>
<th>Parent node path</th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td></td>
</tr>
<tr>
<td>first</td>
<td>&quot;person&quot;</td>
</tr>
<tr>
<td>last</td>
<td>&quot;person&quot;</td>
</tr>
</tbody>
</table>
Die on error

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

Advanced settings

tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Global Variables

Global Variables

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

NB_LINE_REJECTED: the number of rows rejected. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

tMapROjaiOutput executes the action defined on the documents in a given MapR-DB database based on the flow incoming from the preceding component in your Job.

Writing candidate data in a MapR-DB OJAI database

This scenario uses tMapROjaiOutput to write data about some candidates in a MapR-DB OJAI database.

This scenario applies only to a Talend solution with Big Data.

• tFixedFlowInput: it provides the sample data to be written in the database.
• tMapROjaiOutput: it writes the sample data to the database.
Reading the sample candidate data

Procedure

1. Ensure that the MapR client has been properly installed and configured in the machine in which the current job runs.

2. Drop `tFixedFlowInput` and `tMapROjaiOutput` onto the workspace.

3. Connect `tFixedFlowInput` to `tMapROjaiOutput` using a `Row > Main` link and accept the propagation of the schema from `tMapROjaiOutput`.

4. Double-click `tFixedFlowInput` to open its `Component` view.

5. Click the `[... ]` button to open the schema editor.
• The `_id` column exists already because this column was retrieved from `tMapROjaiOutput` in the previous steps to provide the technical IDs of the documents to be stored in a MapR Ojai database. This column is required by `tMapROjaiOutput`.

• Click the `[+]` button to add the other columns and rename them to `firstname`, `lastname`, `sex`, `married`, `age` and `salary`, respectively. The type of the `married` column should be `Boolean`, the type of the `age` column `Integer` and the `salary` column `Float`.

6. Click **OK** to close the schema editor and accept the propagation of the schema.

7. In the **Mode** area, select the **Use Inline table** radio button to display the table in which you add the sample data.

8. Click the `[+]` button to add four rows and enter the sample data as is displayed in the image above.

   This sample data is used for demonstration purposes only.

   The data of the `String` type should be put in double quotation marks and the data of the three other types should be put without any quotation marks.

**Writing the sample data to the MapR OJAI database**

**Procedure**

1. Double-click `tMapROjaiOutput` to open its **Component** view.
2. In the **Table** field, enter the name of the table to be used in your MapR OJAI database.

3. From the **DB Version** list, select the MapR OJAI database version you are using.

   At runtime, tMapROjaiOutput automatically reads the connection information to the database from the `\opt\mapr\conf\mapr-clusters.conf` file of your MapR client.

   In this example, the content of this file reads as follows:

   ```
   mapr521   mapr521:7222
   ```

4. From the **Action on table** drop-down list, select **Drop table if exists and create** and then from the **Action on data** drop-down list, select **Insert**.

5. In the **Mapping** table, the columns from the schema has automatically appears in the **Column** column. In the **Parent field path** column, enter "Name" within double quotation marks in the `firstname` row and the `lastname` row and enter "Status" within double quotation marks in the `sex` row, the `married` row and the `age` row.

   This configuration groups the candidates’ first names and last names in the **Name** field and their sexes, marriage status and ages in the **Status** field in the outputted data flow.

6. Press **F6** to run the Job.
tMapRStreamsCommit

Connects to a given tMapRStreamsInput to perform a consumer offset commit.

tMapRStreamsCommit saves the current state of the tMapRStreamsInput to which it is connected. Note that the term 'commit' in this component means saving what messages are consumed by this tMapRStreamsInput component at the moment of committing.

tMapRStreamsCommit Standard properties

These properties are used to configure tMapRStreamsCommit running in the Standard Job framework. The Standard tMapRStreamsCommit component belongs to the Internet family. The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

| Commit offsets from | Select the tMapRStreamsInput component from which the consumed messages are committed. |

Advanced settings

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

Usage

| Usage rule | Although tMapRStreamsCommit can be used as an end component in a Subjob flow or can be called independently as its own Subjob, it is typically used standalone to commit a batch of offsets in one go. If you need to commit offsets regularly, it is recommended to use the Auto-commit feature in tMapRStreamsInput. |

Related scenarios

No scenario is available for the Standard version of this component yet.
tMapRStreamsConnection

Opens a reusable connection to a given MapR Streams cluster so that the other MapR Streams components can reuse this connection.

**tMapRStreamsConnection Standard properties**

These properties are used to configure tMapRStreamsConnection running in the Standard Job framework.

The Standard tMapRStreamsConnection component belongs to the Big Data family.

The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <em>Built-In</em> or <em>Repository</em>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Built-In</em>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><em>Repository</em>: Select the repository file where the properties are stored.</td>
</tr>
<tr>
<td></td>
<td>The properties are stored centrally under the <em>Hadoop Cluster</em> node of the <em>Repository</em> tree.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distribution and Version</th>
<th>Select the MapR distribution to be used. Only MapR V5.2 onwards is supported by the MapRDB components.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the distribution you need to use with your MapRDB database is not officially supported by this MapRDB component, that is to say, this distribution is MapR but is not listed in the <em>Version</em> drop-down list of this component or this distribution is not MapR at all, select <em>Custom</em>.</td>
</tr>
<tr>
<td></td>
<td>1. Select <em>Import from existing version</em> to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.</td>
</tr>
<tr>
<td></td>
<td>2. Select <em>Import from zip</em> to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.</td>
</tr>
<tr>
<td></td>
<td>In <em>Talend Exchange</em>, members of <em>Talend</em> community have shared some ready-for-use configuration zip files which you can download from this <em>Hadoop configuration</em> list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the <em>Import from existing version</em> option to</td>
</tr>
</tbody>
</table>
take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

Note: In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see Connecting to a custom Hadoop distribution on page 1589.

### NameNode URI
Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called masternode as the NameNode, then the location is hdfs://masternode:portnumber. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS.

### Force MapR ticket authentication
If the MapR cluster to be used is secured with the MapR ticket authentication mechanism, set the MapR ticket authentication configuration by following the explanation in Setting up the MapR ticket authentication on page 1652.

In addition, if your cluster also uses Kerberos to secure your MapR Streams, select the Use Kerberos authentication check box to configure Kerberos for your Job.

- You can also find an example of how to configure Kerberos authentication for a Talend Job in Talend Help Center (https://help.talend.com). Although this example uses Cloudera for demonstration, the operations it describes are generic and thus applicable to MapR as well.

### Advanced settings

| Hadoop properties | Talend Studio uses a default configuration for its engine to perform operations in a Hadoop distribution. If |
you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.

<table>
<thead>
<tr>
<th>tMapRStreamsConnection</th>
<th></th>
</tr>
</thead>
</table>
| you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.  
- Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.  |

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect the log data at a component level.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect the log data at a component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.

**Usage**

**Usage rule**

This component is generally used with other MapR Streams components.

**Prerequisites**

The Hadoop distribution must be properly installed, so as to guarantee the interaction with **Talend Studio**. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under **MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native**. For example, the library for Windows is **lib\native\MapRClient.dll** in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.
Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

Related scenarios

No scenario is available for the Standard version of this component yet.
tMapRStreamsCreateStream

Creates a MapR Streams stream or topic that the other MapR Streams components can use. This component allows you to visually issue the command to create or update a stream or a topic.

**tMapRStreamsCreateStream Standard properties**

These properties are used to configure tMapRStreamsCreateStream running in the Standard Job framework.

The Standard tMapRStreamsCreateStream component belongs to the Internet family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Property type | Either Built-In or Repository.  
|---------------|--------------------------------------------------------------------------------|
|               | **Built-In**: No property data stored centrally.  
|               | **Repository**: Select the repository file where the properties are stored.  
|               | The properties are stored centrally under the Hadoop Cluster node of the Repository tree.  

| tMapRStreamsConnection | Select the tMapRStreamsConnection component to reuse the connection details you already defined. This parameter is mandatory for this component to connect to MapR Streams. |

<table>
<thead>
<tr>
<th>Action</th>
<th>Select how a stream or a topic is created. The related parameters to each action are displayed only when this action is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Create stream</strong>: it creates a stream. If this stream exists already, the entire Job is stopped.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Create stream if not exists</strong>: it creates a stream when this stream does not exist. If this stream exists, the Job skip the creation to move to the next step.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Alter stream</strong>: it changes the configuration of a stream.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Create topic</strong>: it creates a topic. If this topic exists already, the entire Job is stopped.</td>
</tr>
<tr>
<td></td>
<td>Note that the stream to which this topic belongs must already exist.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Create topic if not exists</strong>: it creates a topic when this topic does not exist. If this topic exists, the Job skip the creation to move to the next step.</td>
</tr>
<tr>
<td></td>
<td>Note that the stream to which this topic belongs must already exist.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Alter topic</strong>: it changes the configuration of a topic.</td>
</tr>
<tr>
<td>Stream path</td>
<td>Enter the path to the stream to be created or updated in MapR-FS.</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Stream permissions | Grant the permissions to a user account over the stream to be created or updated. These permissions are about:
- Adminperm
- Consumeperm
- Copyperm
- Produceperm
- Topicperm
The syntax you need to use is `u:username_or_userid_of_the_account` within double quotation marks.
When you have selected *Alter stream* from the *Action* list, a list of check boxes are displayed in the *Stream permissions* area so that you can select the check box corresponding to each permission to be changed and then grant this permission to a different user account.
For details about these permissions, see MapR documentation at MapR Streams Security. |
| Stream settings | Define the configuration of the stream to be created or updated. The parameters to be set are:
- **Automatic topic creation**: select the *Enable automatic topic creation* check box to allow a topic to be automatically created when a producer creates the first message for it and this topic does not already exist.
  For further information about this automatic creation, see MapR documentation at Topic Creation.
- **Default number of partitions**: this parameter defines how many partitions are created in each new topic in a stream.
  For further information about the default number of partitions, see MapR documentation at Default Partitions.
- **Message time to live**: this parameter defines the length of time (in seconds) messages persist in a stream.
  For further information about this message survival time, see MapR documentation at Time-to-live for Messages.
- **Compression**: this is the type of the compression algorithms MapR provides to compress files stored in the MapR cluster.
  For further information about MapR compression, see MapR documentation at Compression. |

When you have selected *Alter stream* from the *Action* list, a list of check boxes are displayed in the *Stream settings* area so that you can select the check box
corresponding to each parameter to be changed and then update this parameter.

<table>
<thead>
<tr>
<th>Topic name</th>
<th>Enter the name of the topic to be created. This field is displayed when you have selected one of the actions about topics from the <strong>Action</strong> list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of partitions</strong> or <strong>Set number of partitions</strong></td>
<td>Enter the number of partitions to be created for the topic. This field is displayed when you have selected one of the actions about topics from the <strong>Action</strong> list.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

**Usage**

| Usage rule | This component is used standalone to create or update a topic or a stream to be used by the other MapR Streams components. |

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)). |

**Related scenarios**

No scenario is available for the Standard version of this component yet.
## tMapRStreamsInput

Transmits messages to the Job that runs transformations over these messages. Only MapR V5.2 onwards is supported by this component.

### tMapRStreamsInput Standard properties

These properties are used to configure tMapRStreamsInput running in the Standard Job framework.

The Standard tMapRStreamsInput component belongs to the Internet family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

#### Basic settings

| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
Note that the schema of this component is read-only. It stores the messages sent from the message producer. |
|----------------------------|--------------------------------------------------------------------------------------------------|
| **Output type**            | Select the type of the data to be sent to the next component.
Typically, using String is recommended, because tMapRStreamsInput can automatically translate the MapR Streams byte[] messages into strings to be processed by the Job. However, in case that the format of MapR Streams messages is not known to tMapRStreamsInput, such as Protobuf, you can select byte[] and then use a Custom code component such as tJavaRow to deserialize the messages into strings so that the other components of the same Job can process these messages. |
| **Use an existing connection** | Select this check box and from the list displayed select the relevant connection component to reuse the connection details you have already defined. |
| **Distribution and Version** | Select the MapR distribution to be used. Only MapR V5.2 onwards is supported by the MapRDB components.
If the distribution you need to use with your MapRDB database is not officially supported by this MapRBD component, that is to say, this distribution is MapR but is not listed in the Version drop-down list of this components or this distribution is not MapR at all, select Custom.

1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide. |
2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see **Connecting to a custom Hadoop distribution** on page 1589.

<table>
<thead>
<tr>
<th>Topic name</th>
<th>Enter the name of the topic from which <strong>tMapRStreamsInput</strong> receives the feed of messages. You must enter the name of the stream to which this topic belongs. The syntax is <code>path_to_the_stream:topic_name</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consumer group ID</td>
<td>Enter the name of the consumer group to which you want the current consumer (the <strong>tMapRStreamsInput</strong> component) to belong. This consumer group will be created at runtime if it does not exist at that moment.</td>
</tr>
<tr>
<td>Reset offsets on consumer group</td>
<td>Select this check box to clear the offsets saved for the consumer group to be used so that this consumer group is handled as a new group that has not consumed any messages.</td>
</tr>
</tbody>
</table>
New consumer group starts from

Select the starting point from which the messages of a topic are consumed.

In MapR Streams, the increasing ID number of a message is called offset. When a new consumer group starts, from this list, you can select beginning to start consumption from the oldest message of the entire topic, or select latest to wait for a new message.

Note that the consumer group takes into account only the offset-committed messages to start from.

Each consumer group has its own counter to remember the position of a message it has consumed. For this reason, once a consumer group starts to consume messages of a given topic, a consumer group recognizes the latest message only with regard to the position where this group stops the consumption, rather than to the entire topic. Based on this principle, the following behaviors can be expected:

- If you are resuming an existing consumer group, this option determines the starting point for this consumer group only if it does not already have a committed starting point. Otherwise, this consumer group starts from this committed starting point. For example, a topic has 100 messages. If an existing consumer group has successfully processed 50 messages, and has committed their offsets, then the same consumer group restarts from the offset 51.

- If you create a new consumer group or reset an existing consumer group, which, in either case, means this group has not consumed any message of this topic, then when you start it from latest, this new group starts and waits for the offset 101.

Auto-commit offsets

Select this check box to make tMapRStreamsInput automatically save its consumption state at the end of each given time interval. You need to define this interval in the Interval field that is displayed.

Note that the offsets are committed only at the end of each interval. If your Job stops in the middle of an interval, the message consumption state within this interval is not committed.

Stop after a maximum total duration (ms)

Select this check box and in the pop-up field, enter the duration (in milliseconds) at the end of which tMapRStreamsInput stops running.

Stop after receiving a maximum number of messages

Select this check box and in the pop-up field, enter the maximum number of messages you want tMapRStreamsInput to receive before it automatically stops running.

Stop after maximum time waiting between messages (ms)

Select this check box and in the pop-up field, enter the waiting time (in milliseconds) by tMapRStreamsInput for a new message. If tMapRStreamsInput does not receive any new message when this waiting time meets its end, it automatically stops running.
### Advanced settings

<table>
<thead>
<tr>
<th>Advanced setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Consumer properties</strong></td>
<td>Add the MapR Streams consumer properties you need to customize to this table. For further information about the consumer properties you can define in this table, see the MapR Streams documentation at <a href="#">MapR Streams Overview</a>.</td>
</tr>
<tr>
<td><strong>Timeout precision(ms)</strong></td>
<td>Enter the time duration in millisecond at the end of which you want a timeout exception to be returned if no message is available for consumption. The value -1 indicates that no timeout is set.</td>
</tr>
<tr>
<td><strong>Load the offset with the message</strong></td>
<td>Select this check box to output the offsets of the consumed messages to the next component. When selecting it, a read-only column called offset is added to the schema.</td>
</tr>
<tr>
<td><strong>Custom encoding</strong></td>
<td>You may encounter encoding issues when you process the stored data. In that situation, select this check box to display the Encoding list. Select the encoding from the list or select Custom and define it manually.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

| Global Variables                                      | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see [Talend Studio User Guide](#). |

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as a start component and requires an output link. When the MapR Streams topic it needs to use does not exist, you can first create this topic using either the tMapRStreamsCreateTopic component or your MapR command-line interface.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prerequisites</strong></td>
<td>The Hadoop distribution must be properly installed, so as to guarantee the interaction with <a href="#">Talend Studio</a>. The</td>
</tr>
</tbody>
</table>
following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

### Related scenarios

No scenario is available for the Standard version of this component yet.
tMapRStreamsOutput

Publishes messages into a MapR Streams system. Only MapR V5.2 onwards is supported by this component.

This component receives messages serialized into byte arrays by its preceding component and issues these messages into a given MapR Streams system.

tMapRStreamsOutput Standard properties

These properties are used to configure tMapRStreamsOutput running in the Standard Job framework.

The Standard tMapRStreamsOutput component belongs to the Internet family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

| **Schema** and **Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
Note that the schema of this component is read-only. It stores the messages to be published. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and from the list displayed select the relevant connection component to reuse the connection details you have already defined.</td>
</tr>
</tbody>
</table>
| **Distribution** and **Version** | Select the MapR distribution to be used. Only MapR V5.2 onwards is supported by the MapRDB components.
If the distribution you need to use with your MapRDB database is not officially supported by this MapRBD component, that is to say, this distribution is MapR but is not listed in the Version drop-down list of this components or this distribution is not MapR at all, select Custom.
1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.
2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.
In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different
Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see [Connecting to a custom Hadoop distribution](#) on page 1589.

<table>
<thead>
<tr>
<th>Topic name</th>
<th>Enter the name of the topic you want to publish messages to. This topic must already exist. You must enter the name of the stream to which this topic belongs. The syntax is <code>path_to_the_stream:topic_name</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compress the data</td>
<td>Select the <strong>Compress the data</strong> check box to compress the output data.</td>
</tr>
</tbody>
</table>

## Advanced settings

| Producer properties | Add the MapR Streams producer properties you need to customize to this table. For further information about the producer configuration you can define in this table, see the section describing the important producer configuration properties for MapR Streams in MapR documentation at [MapR Streams Overview](#). |

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

## Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an *After* variable and it returns a string. This variable functions |
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

#### Usage rule

This component is an end component. It requires a **tJavaRow** or **tJava** component to transform the incoming data into serialized byte arrays.

The following sample shows how to construct a statement to perform this transformation:

```java
output_row.serializedValue = input_row.users.getBytes();
```

In this code, the `output_row` variable represents the schema of the data to be output to **tMapRStreamsOutput** and `output_row.serializedValue` the single read-only column of that schema; the `input_row` variable represents the schema of the incoming data and `input_row.users` the input column called `users` to be transformed to byte arrays by the `getBytes()` method.

#### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

  Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the **[Preferences]** dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to...
make full use of the **Data viewer** to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

### Related scenarios

No scenario is available for the Standard version of this component yet.
tMarketoBulkExec

Imports leads or custom objects into Marketo from a local file in the REST API mode.

**tMarketoBulkExec Standard properties**

These properties are used to configure tMarketoBulkExec running in the Standard Job framework.

The Standard tMarketoBulkExec component belongs to two families: Business and Cloud.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

This property is not available when other connection component is selected from the Connection Component drop-down list.

<table>
<thead>
<tr>
<th>Connection component</th>
<th>Select the component whose connection details will be used to set up the connection to Marketo from the drop-down list.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Endpoint address</th>
<th>Enter the API Endpoint URL of the Marketo Web Service. The API Endpoint URL can be found on the Marketo Admin &gt; Web Services panel.</th>
</tr>
</thead>
</table>

| Client access ID             | Enter the client Id for the access to the Marketo Web Service.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The client Id and client secret can be found on the Marketo Admin &gt; LaunchPoint panel, by clicking View Details for the service to be accessed. You can also contact Marketo Support via <a href="mailto:support@marketo.com">support@marketo.com</a> for further information.</td>
</tr>
</tbody>
</table>

| Secret key                   | Enter the client secret for the access to the Marketo Web Service.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>
### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component.

The schema of this component is predefined and it will be slightly different according to the type of the data (Leads or CustomObjects) to be imported. You can click next to **Edit schema** to view its schema.

- **batchId**: the id of the batch.
- **importId** (for Leads only): the id of the import.
- **importTime** (for CustomObjects only): the time spent on the batch.
- **message**: the status message of the batch.
- **numOfLeadsProcessed/numOfObjectsProcessed**: the number of rows processed so far.
- **numOfRowsFailed**: the number of rows failed so far.
- **numOfRowsWithWarning**: the number of rows with a warning so far.
- **objectApiName** (for CustomObjects only): the API name of the custom object.
- **operation** (for CustomObjects only): the bulk operation type.
- **status**: the status of the batch.
- **failuresLogFile**: the log for the failures.
- **warningsLogFile**: the log for the warnings.

### Import to

Select the type of the data to be imported, either Leads or CustomObjects.

### File format

Select the type of the file containing the data to be imported, csv, tsv or ssv.

### Lookup field

Select the lookup field that will be used for deduplication.

This property is available only when Leads is selected from the **Import to** drop-down list.

### List Id

Specify the ID of the static list to import to.

This property is available only when Leads is selected from the **Import to** drop-down list.

### Partition name

Specify the name of the lead partition to import to.

This property is available only when Leads is selected from the **Import to** drop-down list.

### CustomObject name

Specify the API name of the custom object.

This property is available only when CustomObjects is selected from the **Import to** drop-down list.
### Bulk file path
- Specify the path to the file containing the data to be imported.

### Poll wait time in seconds
- Specify the amount of time (in seconds) for polling for the data.

### Batch log download path
- Specify the path to the batch log file(s) to be downloaded, including the failures log file and the warnings log file.

  **When importing leads**, the failures log file will be named `bulk_leads_<batchId>_failures.csv` and the warnings log file will be named `bulk_leads_<batchId>_warnings.csv`.

  **When importing custom objects**, the failures log file will be named `bulk_customobjects_<customObjectName>_<batchId>_failures.csv` and the warnings log file will be named `bulk_customobjects_<customObjectName>_<batchId>_warnings.csv`.

### Die on error
- Select the check box to stop the execution of the Job when an error occurs.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API Mode</strong></td>
<td>Specify the Marketo API mode. Note that only the REST API mode is available to use for this component. For more information about the Marketo REST API, see <a href="http://developers.marketo.com/documentation/rest/">http://developers.marketo.com/documentation/rest/</a>.</td>
</tr>
<tr>
<td><strong>Timeout</strong></td>
<td>Enter the timeout value (in milliseconds) for the connection to the Marketo Web Service before terminating the attempt.</td>
</tr>
<tr>
<td><strong>Max reconnection attempts</strong></td>
<td>Enter the maximum number of reconnect attempts to the Marketo Web Service before giving up.</td>
</tr>
<tr>
<td><strong>Attempt interval time</strong></td>
<td>Enter the time period (in milliseconds) between subsequent reconnection attempts.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NUMBER_CALL</strong></td>
<td>The number of calls. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>
Usage

| Usage rule | This component can be used as a standalone component or as a start component of a Job or Subjob. |

Related scenario

No scenario is available for this component yet.
tMarketoConnection

Opens a connection to Marketo that can then be reused by other Marketo components.

**tMarketoConnection Standard properties**

These properties are used to configure tMarketoConnection running in the Standard Job framework.
The Standard tMarketoConnection component belongs to two families: Business and Cloud.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

**Endpoint address**

Enter the API Endpoint URL of the Marketo Web Service. The API Endpoint URL can be found on the Marketo Admin > Web Services panel.

**Client access ID**

Enter the client Id for the access to the Marketo Web Service.
The client Id and client secret can be found on the Marketo Admin > LaunchPoint panel, by clicking View Details for the service to be accessed. You can also contact Marketo Support via support@marketo.com for further information.

**Secret key**

Enter the client secret for the access to the Marketo Web Service.
To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**Advanced settings**

| API Mode |
| Select the Marketo API mode, either REST API (default) or SOAP API. |
| For more information about the Marketo REST API, see http://developers.marketo.com/documentation/rest/. |
For more information about the Marketo SOAP API, see http://developers.marketo.com/documentation/soap/.

| **Timeout** | Enter the timeout value (in milliseconds) for the connection to the Marketo Web Service before terminating the attempt. |
| **Max reconnection attempts** | Enter the maximum number of reconnect attempts to the Marketo Web Service before giving up. |
| **Attempt interval time** | Enter the time period (in milliseconds) between subsequent reconnection attempts. |
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| **NUMBER_CALL** | The number of calls. This is an After variable and it returns an integer. |
| **ERROR_MESSAGE** | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

**Usage**

| **Usage rule** | This component is more commonly used with other Marketo components and it opens a connection that can be reused by them. |

**Related scenario**

For a related scenario, see Data transmission with Marketo using REST API on page 2092.
tMarketoInput

Retrieves lead records, activity history, lead changes, and custom object related data from Marketo.

**tMarketoInput Standard properties**

These properties are used to configure tMarketoInput running in the Standard Job framework.

The Standard tMarketoInput component belongs to two families: Business and Cloud.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
<tr>
<td>Connection component</td>
<td>Select the component whose connection details will be used to set up the connection to Marketo from the drop-down list.</td>
</tr>
<tr>
<td>Endpoint address</td>
<td>Enter the API Endpoint URL of the Marketo Web Service. The API Endpoint URL can be found on the Marketo Admin &gt; Web Services panel.</td>
</tr>
<tr>
<td>Client access ID</td>
<td>Enter the client Id for the access to the Marketo Web Service.</td>
</tr>
<tr>
<td></td>
<td>The client Id and client secret can be found on the Marketo Admin &gt; LaunchPoint panel, by clicking View Details for the service to be accessed. You can also contact Marketo Support via <a href="mailto:support@marketo.com">support@marketo.com</a> for further information.</td>
</tr>
<tr>
<td>Secret key</td>
<td>Enter the client secret for the access to the Marketo Web Service.</td>
</tr>
<tr>
<td></td>
<td>To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
</tbody>
</table>
**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

- **Built-In:** You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

- **Repository:** You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

The schema of this component is predefined and it will be different according to the API mode selected, the operation chosen or the custom object specified.

Note that when the API mode is changed, you need to refresh the schema columns manually by switching back and forth the components in the **Connection Component** drop-down list (only when reusing the connection created by a `tMarketoConnection` component) and changing the operation to be performed in the **Operation** drop-down list.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema:** choose this option to view the schema only.

- **Change to built-in property:** choose this option to change the schema to **Built-in** for local changes.

- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

Note that the dynamic schema feature for this component is only supported in the REST API mode.

**Operation**

Select one of the following operations to be performed from the drop-down list.

- **getLead:** This operation retrieves a single lead record.

- **getMultipleLeads:** This operation retrieves a batch of lead records that match the specified lead selector search condition.

- **getLeadActivity:** This operation retrieves the activity history that matches the specified search condition.

- **getLeadChanges:** This operation retrieves data value changes and new lead activities after a given time.
- **CustomObject**: Select this item to perform specific action on a custom object. This item is available only in the REST API mode.

### Mappings

Complete this table to map the column names of the component schema with the column names in Marketo. By default, the column names in **Column name** fields are the same as what they are in the schema.

Note that because some column names in Marketo may contain blank space, which is not allowed in the component schema, you need to specify the corresponding column fields in the **Marketo column name** column. If the defined column names in schema are the same as column names in Marketo, it is not necessary to set the columns mapping.

This table is not available when **CustomObject** is selected from the **Operation** list in the REST API mode.

### Lead selector

Select one of the following lead selector types from the drop-down list.

- **LeadKeySelector**: Select this type to retrieve lead records that are identified by the specified lead key.

- **StaticListSelector**: Select this type to retrieve lead records that are members of the given static list. With this option selected, you need to specify the static list by its name or id. To do this, select **STATIC_LIST_NAME** or **STATIC_LIST_ID** from the **List param** drop-down list and then enter the key value of the name or id in the **List param value** field.

- **LastUpdateAtSelector**: Select this type to retrieve lead records that were updated on specific dates. This item is available only in the SOAP API mode.

This list is available only when **getMultipleLeads** is selected from the **Operation** list.

### Lead key type

Select a field by which you want to query the lead from the drop-down list. This list is available when:

- **getLead** is selected from the **Operation** list, or
- **getMultipleLeads** is selected from the **Operation** list and **LeadKeySelector** is selected from the **Lead selector** list, or
- **getLeadActivity** is selected from the **Operation** list in the SOAP API mode.

### Lead key value

Specify the value of the field by which you want to query the lead. This field is available when:

- **getLead** is selected from the **Operation** list, or
- **getLeadActivity** is selected from the **Operation** list in the SOAP API mode.

### Lead key values

Specify the list of key values. This field is available only when **getMultipleLeads** is selected from the **Operation** list and
LeadKeySelector is selected from the **Lead Selector** list.

### Set Include Types

Select this check box and in the **Activity** table displayed, specify the types of activity you want to retrieve by clicking the [+] button to add as many rows as needed, each row for an activity type, then clicking the right side of each cell and selecting an activity type from the drop-down list. This check box is available when:

- `getLeadActivity` is selected from the **Operation** list, or
- `getLeadChanges` is selected from the **Operation** list in the SOAP API mode.

### Set Exclude Types

Select this check box and in the **Activity** table displayed, specify the types of activity you do not want to retrieve by clicking the [+] button to add as many rows as needed, each row for an activity type, then clicking the right side of each cell and selecting an activity type from the drop-down list. This check box is available when:

- `getLeadActivity` is selected from the **Operation** list, or
- `getLeadChanges` is selected from the **Operation** list in the SOAP API mode.

### Oldest update date

Specify the time of the earliest update to retrieve only the leads updated since the specified time.

It supports the time format that Marketo supports. For the supported time format, see [http://developers.marketo.com/documentation/soap/getmultipleleads/](http://developers.marketo.com/documentation/soap/getmultipleleads/).

This field is available only in the SOAP API mode when `getMultipleLeads` is selected from the **Operation** list and `LastUpdateAtSelector` is selected from the **Lead selector** list.

### Latest update date

Specify the time of the latest update to retrieve only the leads updated before the specified time.

It supports the time format that Marketo supports. For the supported time format, see [http://developers.marketo.com/documentation/soap/getmultipleleads/](http://developers.marketo.com/documentation/soap/getmultipleleads/).

This field is available only in the SOAP API mode when `getMultipleLeads` is selected from the **Operation** list and `LastUpdateAtSelector` is selected from the **Lead selector** list.

### Oldest create date

Type in the time of the earliest creation to retrieve only the data since the specified time. It supports the time format that Marketo supports.

This field is available only in the SOAP API mode when `getLeadChanges` is selected from the **Operation** list.

### Latest creation date

Type in the time of the latest creation to retrieve only the data before the specified time. It supports the time format that Marketo supports.
This field is available only in the SOAP API mode when `getLeadChanges` is selected from the **Operation** list.

Note that **Oldest create date** and **Latest creation date** can be specified together or separately.

<table>
<thead>
<tr>
<th>Field list</th>
<th>Enter the comma separated list of field names. This field is available when <code>getLeadChanges</code> is selected from the <strong>Operation</strong> list in the REST API mode.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since DateTime</td>
<td>Specify the timestamp used to filter results to only include leads with activity since then. This field is available when <code>getLeadActivity</code> or <code>getLeadChanges</code> is selected from the <strong>Operation</strong> list in the REST API mode.</td>
</tr>
</tbody>
</table>
| Action | Select the operation to be performed on the specified custom object.  
- **describe**: Returns the metadata regarding a given custom object.  
- **list**: Returns a list of custom object types available in the target instance, along with id and deduplication information for each type. With this option selected, you need to specify a comma-separated list of names to filter types in the **CustomObject names** field.  
- **get**: Retrieves a list of custom objects records based on filter and set of values. This list is available only when **CustomObject** is selected from the **Operation** drop-down list. |
| CustomObject name | Specify the name of the custom object. This field is available only when **CustomObject** is selected from the **Operation** list and **describe** or **get** is selected from the **Action** list. |
| Fetch schema | Click this button to retrieve the schema of the specified custom object. This button is available only when **CustomObject** is selected from the **Operation** list and **get** is selected from the **Action** list. |
| Filter type | Specify the field to filter on. Searchable fields can be retrieved by performing a describe action with the custom object. This field is available only when **CustomObject** is selected from the **Operation** list and **get** is selected from the **Action** list. |
| Filter values | Specify a comma-separated list of field values to match against. This field is available only when **CustomObject** is selected from the **Operation** list and **get** is selected from the **Action** list. |
**Batch size**

Specify the number of records to be processed in each batch.

This field is available when:

- `getMultipleLeads` is selected from the **Operation** list and `Selector` or `LastUpdateAtSelector` is selected from the **Lead Selector** list.
- `getLeadActivity` or `getLeadChanges` is selected from the **Operation** list.
- `CustomObject` is selected from the **Operation** list and `get` is selected from the **Action** list.

**Die on error**

Select the check box to stop the execution of the Job when an error occurs.

### Advanced settings

**API Mode**

Select the Marketo API mode, either REST API (default) or SOAP API.

For more information about the Marketo REST API, see [http://developers.marketo.com/documentation/rest/](http://developers.marketo.com/documentation/rest/).

For more information about the Marketo SOAP API, see [http://developers.marketo.com/documentation/soap/](http://developers.marketo.com/documentation/soap/).

**Timeout**

Enter the timeout value (in milliseconds) for the connection to the Marketo Web Service before terminating the attempt.

**Max reconnection attempts**

Enter the maximum number of reconnect attempts to the Marketo Web Service before giving up.

**Attempt interval time**

Enter the time period (in milliseconds) between subsequent reconnection attempts.

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**NUMBER_CALL**

The number of calls. This is an After variable and it returns an integer.

**ERROR_MESSAGE**

The error message generated by the component when an error occurs. This is an After variable and it returns a string.

### Usage

**Usage rule**

This component is usually used as a start component of a Job or Subjob and it always needs an output link.
Related Scenario

For a related use case, see Data transmission with Marketo using REST API on page 2092.
tMarketoListOperation

Adds/removes one or more leads to/from a list in Marketo. Also, it helps you verify the existence of one or more leads in a list in Marketo.

**tMarketoListOperation Standard properties**

These properties are used to configure tMarketoListOperation running in the Standard Job framework. The Standard tMarketoListOperation component belongs to two families: Business and Cloud. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Connection component</strong></td>
<td>Select the component whose connection details will be used to set up the connection to Marketo from the drop-down list.</td>
</tr>
<tr>
<td><strong>Endpoint address</strong></td>
<td>Enter the API Endpoint URL of the Marketo Web Service. The API Endpoint URL can be found on the Marketo Admin &gt; Web Services panel.</td>
</tr>
<tr>
<td><strong>Client access ID</strong></td>
<td>Enter the client Id for the access to the Marketo Web Service.</td>
</tr>
<tr>
<td></td>
<td>The client Id and client secret can be found on the Marketo Admin &gt; LaunchPoint panel, by clicking View Details for the service to be accessed. You can also contact Marketo Support via <a href="mailto:support@marketo.com">support@marketo.com</a> for further information.</td>
</tr>
<tr>
<td><strong>Secret key</strong></td>
<td>Enter the client secret for the access to the Marketo Web Service.</td>
</tr>
<tr>
<td></td>
<td>To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>
### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

Note that the dynamic schema feature for this component is only supported in the REST API mode.

### Operation

Select an operation to be performed from the drop-down list.

- **addTo**: This operation adds one or more leads to a list.
- **isMemberOf**: This operation verifies whether the specific leads exist in the list.
- **removeFrom**: This operation removes one or more leads from a list.

### Add or remove many prospects

Select this check box to add multiple leads to or remove multiple leads from a list.

This check box is available only when **addTo** or **removeFrom** is selected from the **Operation** list.

### Die on error

Select the check box to stop the execution of the Job when an error occurs.

Clear the check box to skip any rows on error and complete the process for error-free rows.

When errors are skipped, you can collect the rows on error using a **Row > Reject** connection if the **Add or remove many prospects** check box is cleared.

### Advanced settings

#### API Mode

Select the Marketo API mode, either REST API (default) or SOAP API.

For more information about the Marketo REST API, see [http://developers.marketo.com/documentation/rest/](http://developers.marketo.com/documentation/rest/).

For more information about the Marketo SOAP API, see [http://developers.marketo.com/documentation/soap/](http://developers.marketo.com/documentation/soap/).

#### Timeout

Enter the timeout value (in milliseconds) for the connection to the Marketo Web Service before terminating the attempt.
Max reconnection attempts | Enter the maximum number of reconnect attempts to the Marketo Web Service before giving up.
---|---
Attempt interval time | Enter the time period (in milliseconds) between subsequent reconnection attempts.
StatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Global Variables

| NUMBER_CALL | The number of calls. This is an After variable and it returns an integer.
| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string.

Usage

| Usage rule | This component is usually used as an intermediate component, and it requires an input component and an output component.

Adding a lead record to a Marketo list using SOAP API

The following scenario describes a three-component Job that adds a lead record into a specified list in Marketo using SOAP API.

![Diagram of three components: tFixedFlowInput, tMarketoListOperation, tLogRow]

Setting up the Job for adding a lead record to a Marketo list

Procedure

1. Create a new Job and add a **tFixedFlowInput** component, a **tMarketoListOperation** component, and a **tLogRow** component by typing their names in the design workspace or dropping them from Palette.
2. Double-click the **tMarketoListOperation** component and on the Component tab click Advanced settings to open its advanced settings view, then select SOAP from the API Mode drop-down list.
3. Connect **tFixedFlowInput** to **tMarketoListOperation** using a **Row > Main** connection. In the pop-up dialog box, click Yes to get the schema from the target component.
4. Connect **tMarketoListOperation** to **tLogRow** using a **Row > Main** connection.
Configuring the Job for adding a lead record to a Marketo list

Procedure

1. Double-click the `tFixedFlowInput` component to open its Basic settings view.

   ![tFixedFlowInput_1](image)

   - **Basic settings**
     - **Schema**: Built-In
     - **Mode**: Use Inline Table
     - **Number of rows**: 1
     - **ListKeyType**: MKTOLISTNAME
     - **ListKeyValue**: Test_Comp_QA
     - **LeadKeyType**: IDNUM
     - **LeadKeyValue**: 9300203

2. Click the [...] button next to Edit schema to view its schema.

   ![Schema of tFixedFlowInput_1](image)

3. In the **Mode** area, select Use Inline Table, and then click the [+ ] button to add one row and fill the row with respective values. In this example, these values are: MKTOLISTNAME for ListKeyType, Test_Comp_QA for ListKeyValue, IDNUM for LeadKeyType and 9300203 for LeadKeyValue.

4. Double-click the `tMarketoListOperation` component to open its Basic settings view.

   ![tMarketoListOperation_1](image)
5. In the **Endpoint address** field, enter the Marketo SOAP API endpoint URL that can be found on the Marketo Admin > Web Services panel.

6. In the **Client Access ID** and **Secret key** fields, enter the user ID and encryption key required to access Marketo.

7. From the **Operation** list, select **addTo** to add the lead with the ID 9300203 into the list Test_Comp_QA.

8. Double-click the **tLogRow** component to open its **Basic settings** view.

9. Click the **Sync columns** button to synchronize the schema with the input schema.

10. In the **Mode** area, select **Table (print values in cells of a table)** for better readability of the result.

**Executing the Job to add a lead record to a Marketo list**

**Procedure**

1. Press **Ctrl + S** to save the Job.

2. Press **F6** or click **Run** on the Run tab to run the Job.

   ![Statistics connecting to socket on port 3520](image)
   ![tLogRow_1](image)
   ![ListKeyType | ListKeyValue | LeadKeyType | LeadKeyValue | Success](image)
   ![MKTOLISTNAME | Test_Comp_QA | ID | NUM | 9300203 | true](image)

   As shown above, the result of the operation is displayed on the **Run** console.

3. Double-check the result in Marketo.

![Lead Database](image)

The lead with the ID 9300203 in Marketo is added into the list Test_Comp_QA.
tMarketoOutput

Writes lead records or custom object records from the incoming data flow into Marketo.

**tMarketoOutput Standard properties**

These properties are used to configure tMarketoOutput running in the Standard Job framework.

The Standard tMarketoOutput component belongs to two families: Business and Cloud.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-In: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>Repository: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

This property is not available when other connection component is selected from the Connection Component drop-down list.

<table>
<thead>
<tr>
<th>Connection component</th>
<th>Select the component whose connection details will be used to set up the connection to Marketo from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Endpoint address: Enter the API Endpoint URL of the Marketo Web Service. The API Endpoint URL can be found on the Marketo Admin &gt; Web Services panel.</td>
</tr>
<tr>
<td></td>
<td>Client access ID: Enter the client Id for the access to the Marketo Web Service.</td>
</tr>
<tr>
<td></td>
<td>The client Id and client secret can be found on the Marketo Admin &gt; LaunchPoint panel, by clicking View Details for the service to be accessed. You can also contact Marketo Support via <a href="mailto:support@marketo.com">support@marketo.com</a> for further information.</td>
</tr>
<tr>
<td></td>
<td>Secret key: Enter the client secret for the access to the Marketo Web Service.</td>
</tr>
<tr>
<td></td>
<td>To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>
### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](#).

- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](#).

The schema of this component is predefined and it will be different according to the API mode selected, the operation chosen or the custom object specified.

Note that when the API mode is changed, you need to refresh the schema columns manually by switching back and forth the components in the **Connection Component** drop-down list (only when reusing the connection created by a `tMarketoConnection` component) and changing the operation to be performed in the **Operation** drop-down list.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.

- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

Click **Sync columns** to retrieve the schema from the previous component connected in the **Job**.

Note that the dynamic schema feature for this component is only supported in the REST API mode.

### Operation

Select an operation that allows you to synchronize data between Marketo and another external system.

- **syncLead**: Insert or update (upsert) a single lead record.

- **syncMultipleLeads**: Insert or update (upsert) multiple lead records.

- **deleteLeads** (for REST only): Delete a list of leads.

- **syncCustomObjects** (for REST only): Insert, update, or upsert custom object records to the target instance.

- **deleteCustomObjects** (for REST only): Delete a set of custom object records.
| **Operation type** | Select a type of sync operation to be performed from the drop-down list.  
• **createOnly**: creates new leads.  
• **updateOnly**: updates existing leads.  
• **createOrUpdate**: creates new leads, or updates the leads if they already exist.  
• **createDuplicate**: creates new leads. If they already exist, then creates duplicated leads.  
This list is available only in the REST API mode when **syncLead** or **syncMultipleLeads** is selected from the **Operation** list. |
| **Lookup field** | Select the lookup field that will be used to find the duplicate.  
This list is available only in the REST API mode when **syncLead** or **syncMultipleLeads** is selected from the **Operation** list. |
| **Mappings** | Complete this table to map the column names of the component schema with the column names in Marketo.  
By default, the column names in **Column name** fields are the same as what they are in the schema.  
Note that because some column names in Marketo may contain blank space, which is not allowed in the component schema, you need to specify the corresponding column fields in the **Marketo column name** column. If the defined column names in schema are the same as column names in Marketo, it is not necessary to set the columns mapping.  
This table is available only in the SOAP API mode. |
| **Dedupe prospects record using the email address** | Select this check box to de-duplicate records using email address.  
Clear this check box to create another record which contains the same email address. |
| **Delete leads in batch** | Select this check box to delete the leads in batch.  
This check box is available only when **deleteLeads** is selected from the **Operation** list. |
| **Batch Size** | Specify the number of records to be processed in each batch.  
This field is available when:  
• **syncMultipleLeads** is selected from the **Operation** list, or  
• **deleteLeads** is selected from the **Operation** list and the **Delete leads in batch** check box is selected. |
| **Sync method** | Select a type of sync operation to be performed from the drop-down list. |
### Advanced settings

| **API Mode** | Select the Marketo API mode, either **REST API** (default) or **SOAP API**.  
For more information about the Marketo REST API, see [http://developers.marketo.com/documentation/rest/](http://developers.marketo.com/documentation/rest/).  
For more information about the Marketo SOAP API, see [http://developers.marketo.com/documentation/soap/](http://developers.marketo.com/documentation/soap/). |
| **Timeout** | Enter the timeout value (in milliseconds) for the connection to the Marketo Web Service before terminating the attempt. |
| **Max reconnection attempts** | Enter the maximum number of reconnect attempts to the Marketo Web Service before giving up. |
| **Attempt interval time** | Enter the time period (in milliseconds) between subsequent reconnection attempts. |
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |
Global Variables

| NUMBER_CALL | The number of calls. This is an After variable and it returns an integer. |
| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

Usage

| Usage rule | This component is usually used as an end component of a Job or Subjob and it always needs an input link. |

Data transmission with Marketo using REST API

Here is an example of using Talend components to connect to Marketo, insert several lead records into Marketo, then retrieve these records from Marketo and display them on the console.

Setting up the Job for data transmission with Marketo using REST API

Procedure

1. Add a tMarketoConnection component, two tFixedFlowInput components, a tMarketoOutput component, a tMarketoInput component, and a tLogRow component by typing their names on the design workspace or dropping them from the Palette.
2. Make sure REST is selected from the API Mode drop-down list on the Advanced settings view of tMarketoOutput and tMarketoinput.
3. Connect the first tFixedFlowInput component to the tMarketoOutput component using a Row > Main connection. In the pop-up dialog box, click Yes so that the first tFixedFlowInput component can get the schema from the target tMarketoOutput component.
4. Connect the second tFixedFlowInput component to the tMarketoInput component using a Row > Main connection. In the pop-up dialog box, click No and the second tFixedFlowInput component will not get the schema from the target tMarketoInput component.
5. Connect the `tMarketoInput` component to the `tLogRow` component using a `Row > Main` connection.

6. Connect the `tMarketoConnection` component to the first `tFixedFlowInput` component using a `Trigger > On Subjob Ok` connection.

7. Do the same to connect the first `tFixedFlowInput` component to the second `tFixedFlowInput` component.

### Connecting to Marketo using REST API

#### Before you begin

Before connecting to Marketo using the REST API endpoint, you need to make sure a custom service, which provides credentials to authenticate with Marketo and allows you to access the data in Marketo, has been created and associated with a single API-only user by your Marketo administrator. For more information about the Marketo Custom Service, see [Custom Services](#).

#### Procedure

1. Double-click the `tMarketoConnection` component to open its **Basic settings** view.

   ![Image of tMarketoConnection settings](image)

   - In the **Endpoint address** field, enter the REST API endpoint URL that can be found on the Marketo `Admin > Web Services` panel.
   - In the **Client Access ID and Secret key** fields, enter the client Id and client secret required to access Marketo.

   The client Id and client secret can be found on the Marketo `Admin > LaunchPoint` panel, by clicking **View Details** for the service to be accessed.

### Inserting lead records into Marketo using REST API

#### Procedure

1. Double-click the first `tFixedFlowInput` component to open its **Basic settings** view.

   •
2. Click the [...] button next to Edit schema to open the schema dialog box.

3. In the schema dialog box, select the id and Status columns that don't need input values and click the ✗ button to remove them. Then click OK to save the settings and close the schema dialog box.

4. In the Mode area, select Use Inline Content(delimited file) and enter the following input data in the Content field.

   dev@talend.com;Peter;Wang
   doc@talend.com;Judy;Zhang
   dev@talend.com;Nick;Liu
   dev@talend.com;Mark;Zhao
   doc@talend.com;Alex;Sun

5. Double-click the tMarketoOutput component to open its Basic settings view.
6. From the **Connection Component** drop-down list, select the component whose connection details will be used to set up the connection to Marketo, tMarketoConnection_1 in this example.

7. From the **Operation** drop-down list, select **SyncMultipleLeads** to insert multiple lead records.

8. From the **Operation type** drop-down list, select **createDuplicate** to allow duplicated leads to be created.

### Retrieving lead records from Marketo using REST API

**Procedure**

1. Double-click the second **tFixedFlowInput** component to open its **Basic settings** view.

2. Click the button next to **Edit schema** and in the schema dialog box displayed, define the schema by adding one column `mail` of **String** type. Then click **OK** to save the changes and close the schema dialog box.
3. In the **Mode** area, select **Use Inline Table**. Then click the button twice below the table displayed to add two rows and fill each row with respective values. In this example, two email values dev@talend.com and doc@talend.com are added so that the lead records identified by these email values can be retrieved.

4. Double-click the **tMarketoInput** component to open its **Basic settings** view.

5. From the **Connection Component** drop-down list, select the component whose connection details will be used to set up the connection to Marketo, **tMarketoConnection_1** in this example.

6. Select **getMultipleLeads** from the **Operation** drop-down list to retrieve a batch of lead records.

7. Select **LeadKeySelector** from the **Lead selector** drop-down list, and select **email** from the **Lead key type** list and **email** from the **Lead key values** list to retrieve the leads identified by the email values passed from the second **tFixedFlowInput** component.

8. Double-click the **tLogRow** component to open its **Basic settings** view.

9. In the **Mode** area, select **Table (print values in cells of a table)** for better readability of the result.
Executing the Job to transmit data with Marketo using REST API

Procedure

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to execute the Job.

As shown above, the inserted lead records in Marketo are retrieved and displayed on the console.
**tMarkLogicBulkLoad**

Imports local files into a MarkLogic server database in bulk mode using the MarkLogic Content Pump (MLCP) tool.

**tMarkLogicBulkLoad Standard properties**

These properties are used to configure tMarkLogicBulkLoad running in the Standard Job framework.

The Standard tMarkLogicBulkLoad component belongs to the Big Data family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either <strong>Built-In</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see **Talend Studio User Guide**.

<table>
<thead>
<tr>
<th>Host</th>
<th>Enter the IP address or hostname of the MarkLogic server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Enter the listening port number of the MarkLogic server.</td>
</tr>
<tr>
<td>Database</td>
<td>Enter the name of the MarkLogic database you want to use.</td>
</tr>
</tbody>
</table>
### Username and Password
Enter the user authentication data to access the MarkLogic database.
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

### Load Folder
Specify the directory to the local files to be imported into the MarkLogic database.

### DocId Prefixed As
Specify the prefix used to construct the URI of the imported files. For more information about the URI construction, see [https://docs.marklogic.com/guide/ingestion/content-pump#id_14982](https://docs.marklogic.com/guide/ingestion/content-pump#id_14982).

### Advanced settings

#### Additional MLCP Parameters
Specify additional options for the MLCP import command. For more information about the import options, see [https://docs.marklogic.com/guide/ingestion/content-pump#id_63999](https://docs.marklogic.com/guide/ingestion/content-pump#id_63999).

#### Use external MLCP process
As the MarkLogic Content Pump (MLCP) program is required on the machine where you run your Job that involves this component, you can do either of the following:
- Keep this check box clear to use a built-in MLCP library when running this component to load data. In this case, your Job is memory demanding.
- If you need to use your own MLCP, that is to say, an MLCP external to the Studio, select this check box. In this case, your Job itself is not memory demanding but your MLCP process is.

For further information about MLCP, see the related Marklogic documentation: [Loading Content Using Marklogic Content Pump](https://docs.marklogic.com/guide/ingestion/content-pump#id_63999).

#### tStatCatcher Statistics
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used together with the tMarkLogicInput component to check if the files are imported as expected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Limitation</td>
<td>The MarkLogic Content Pump (MLCP) tool needs to be installed on the machine where you run the Job that involves this component.</td>
</tr>
</tbody>
</table>

### Related scenario

No scenario is available for the Standard version of this component yet.
tMarkLogicClose

Closes an active connection to a MarkLogic database to release the occupied resources.

**tMarkLogicClose Standard properties**

These properties are used to configure tMarkLogicClose running in the Standard Job framework.

The Standard tMarkLogicClose component belongs to the Big Data family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Component List</th>
<th>Select the tMarkLogicConnection component that opens the connection you need to close from the list.</th>
</tr>
</thead>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</th>
</tr>
</thead>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other MarkLogic components, especially with the tMarkLogicConnection component.</th>
</tr>
</thead>
</table>

| Dynamic settings | Click the [*] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having |
the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenario

No scenario is available for the Standard version of this component yet.
tMarkLogicConnection

Opens a connection to a MarkLogic database that can then be reused by other MarkLogic components.

**tMarkLogicConnection Standard properties**

These properties are used to configure tMarkLogicConnection running in the Standard Job framework.

The Standard tMarkLogicConnection component belongs to the Big Data family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

| Host | Enter the IP address or hostname of the MarkLogic server. |
| Port | Enter the listening port number of the MarkLogic server. |
| Database | Enter the name of the MarkLogic database you want to use. |
| Username and Password | Enter the user authentication data to access the MarkLogic database. |
| Authentication Type | Select an authentication type from the list, either DIGEST or BASIC. For more information about the types of authentication, see [http://docs.marklogic.com/guide/security/authentication#id_56266](http://docs.marklogic.com/guide/security/authentication#id_56266). |

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After |
variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

| Usage rule | This component is more commonly used with other MarkLogic components. |

Related scenario

No scenario is available for the Standard version of this component yet.
tMarkLogicInput

Searches document content in a MarkLogic database based on a string query.
tMarkLogicInput passes on the document URI and content to the next component via a Row > Main connection.

tMarkLogicInput Standard properties

These properties are used to configure tMarkLogicInput running in the Standard Job framework.
The Standard tMarkLogicInput component belongs to the Big Data family.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

Use an existing connection

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

Host

Enter the IP address or hostname of the MarkLogic server.

Port

Enter the listening port number of the MarkLogic server.
| **Database** | Enter the name of the MarkLogic database you want to use. |
| **Username and Password** | Enter the user authentication data to access the MarkLogic database.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Authentication** | Select an authentication type from the list, either DIGEST or BASIC. For more information about the types of authentication, see [http://docs.marklogic.com/guide/security/authentication#id_56266](http://docs.marklogic.com/guide/security/authentication#id_56266). |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
The schema of this component is read-only. You can click the [...] button next to Edit schema to view the predefined schema that contains the following two columns:  
• docId: the URI of the document.  
• docContent: the content of the document. |
| **Query Criteria** | Specify the string query based on which the search is performed. For more information about the string query, see [https://docs.marklogic.com/guide/search-dev/string-query](https://docs.marklogic.com/guide/search-dev/string-query). |

### Advanced settings

| **Maximal Retrieve Number** | Enter the maximum number of documents to be retrieved. |
| **Use Query Option** | Select this check box to customize the query with a query option. The query option will be stored on the REST server and can then be referenced by name by other queries. For more information about the query options, see [http://docs.marklogic.com/guide/java/query-options#id_20346](http://docs.marklogic.com/guide/java/query-options#id_20346).  
**Query Literal Type**: select a type of presentation from which the query option is created, either XML or JSON.  
**Query Option Name**: specify the name of the query option.  
**Query Option Literals**: implement the query option from a raw XML or JSON representation according to the selected representation type. |
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |
Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.
For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component is usually used as a start component and it needs an output flow. |

Dynamic settings

| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.
The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

Related scenario

No scenario is available for the Standard version of this component yet.
tMarkLogicOutput

Creates, updates or deletes document content in a MarkLogic database.

tMarkLogicOutput allows you to maintain document content in a MarkLogic database according to the predefined operation and the input flow from the preceding component.

**tMarkLogicOutput Standard properties**

These properties are used to configure tMarkLogicOutput running in the Standard Job framework.

The Standard tMarkLogicOutput component belongs to the Big Data family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>• Repository: Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. |

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<p>| Host | Enter the IP address or hostname of the MarkLogic server. |
| Port | Enter the listening port number of the MarkLogic server. |</p>
<table>
<thead>
<tr>
<th>Database</th>
<th>Enter the name of the MarkLogic database you want to use.</th>
</tr>
</thead>
</table>
| Username and Password    | Enter the user authentication data to access the MarkLogic database.  
  To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Authentication           | Select an authentication type from the list, either DIGEST or BASIC. For more information about the types of authentication, see http://docs.marklogic.com/guide/security/authentication#id_56266. |
| Action                   | Select an operation to be performed:  
  • UPSERT: create documents if they do not exist or update the content of existing documents.  
  • PATCH: perform a partial update to the content of the documents.  
  • DELETE: delete documents corresponding to the input flow.  
  Note that when DELETE is selected from the Action list, the input schema should contain one column docId that describes the URI of the documents to be deleted, and other columns will be ignored if there is any. |
| Schema and Edit schema   | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
  The schema of this component is read-only. You can click the [...] button next to Edit schema to view the predefined schema that contains the following two columns:  
  • docId: the URI of the document.  
  • docContent: the content of the document. |

**Advanced settings**

<table>
<thead>
<tr>
<th>Doc Type</th>
<th>Select the type of the documents to be processed: MIXED, PLAIN TEXT, JSON, XML, or BINARY.</th>
</tr>
</thead>
</table>
| Auto Generate Doc ID      | Select this check box to generate automatically the document URIs and in the Doc Id Prefix field displayed enter the prefix used to construct the document URIs.  
  This check box is available only when UPSERT is selected from the Action list and MIXED is not selected from the Doc Type list.  
  • If this check box is selected, the input schema should contain one column docContent that describes the
document content, and other columns will be ignored if there is any.

- If this check box is clear, the input schema should contain two columns `docId` and `docContent` that describe the document URI and the document content correspondingly.

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is usually used as an end component and it always needs an input flow.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Dynamic settings</th>
<th>Click the [*] button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The <strong>Dynamic settings</strong> table is available only when the <strong>Use an existing connection</strong> check box is selected in the <strong>Basic settings</strong> view. Once a dynamic parameter is defined, the <strong>Component List</strong> box in the <strong>Basic settings</strong> view becomes unusable.</td>
</tr>
<tr>
<td></td>
<td>For examples on using dynamic parameters, see <em>Scenario: Reading data from databases through context-based dynamic connections</em> on page 2497 and <em>Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters</em> on page 489.</td>
</tr>
<tr>
<td></td>
<td>For more information on <strong>Dynamic settings</strong> and context variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>
Related scenario

No scenario is available for the Standard version of this component yet.
tMaxDBInput

Reads a database and extracts fields based on a query.

tMaxDBInput executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Main row link.

**tMaxDBInput Standard properties**

These properties are used to configure tMaxDBInput running in the Standard Job framework.

The Standard tMaxDBInput component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Host name</th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
Repository: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Table name</th>
<th>Type in the table name.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query type and Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Guess Query</strong></td>
<td>Click the <strong>Guess Query</strong> button to generate the query which corresponds to your table schema in the Query field.</td>
</tr>
<tr>
<td><strong>Guess schema</strong></td>
<td>Click the <strong>Guess schema</strong> button to retrieve the table schema.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>Trim all the String/Char columns</th>
<th>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trim column</strong></td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>QUERY</strong>: the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>This component requires installation of its related jar files.</td>
</tr>
</tbody>
</table>

**Related scenario**

For a related scenario, see:
tMaxDBOutput

Writes, updates, makes changes or suppresses entries in a database.
tMaxDBOutput executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the job.

**tMaxDBOutput Standard properties**

These properties are used to configure tMaxDBOutput running in the Standard Job framework.
The Standard tMaxDBOutput component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.
For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

- **Host**
  Database server IP address.

- **Port**
  Listening port number of DB server.

- **Database**
  Name of the database.

- **Username and Password**
  DB user authentication data.
  To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

- **Table**
  Name of the table to be written. Note that only one table can be written at a time and that the table must exist for the insert operation to succeed.

- **Action on table**
  On the table defined, you can perform one of the following operations:
  - **None**: No operation is carried out.
  - **Drop and create table**: The table is removed and created again.
| Action on data | Create table: The table does not exist and gets created.  
Create table if not exists: The table is created if it does not exist.  
Clear table: The table content is deleted.  
Truncate table: The table content is deleted. You do not have the possibility to rollback the operation. |
|---|---|
| On the data of the table defined, you can perform:  
Insert: Add new entries to the table. If duplicates are found, job stops.  
Update: Make changes to existing entries  
Insert or update: Insert a new record. If the record with the given reference already exists, an update would be made.  
Update or insert: Update the record with the given reference. If the record does not exist, a new record would be inserted.  
Delete: Remove entries corresponding to the input flow.  
Warning: It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking Edit Schema and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the Advanced settings view where you can simultaneously define primary keys for the Update and Delete operations. To do that: Select the Use field options check box and then in the Key in update column, select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the Key in delete column for the Delete operation. |
| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.  
Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.  
When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. |
You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Die on error**

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

### Advanced settings

<table>
<thead>
<tr>
<th>Commit every</th>
<th>Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional Columns</strong></td>
<td>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Type in the name of the schema column to be altered or inserted as new column</td>
</tr>
<tr>
<td><strong>SQL expression</strong></td>
<td>Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td>Select <strong>Before</strong>, <strong>Replace</strong> or <strong>After</strong> following the action to be performed on the reference column.</td>
</tr>
<tr>
<td><strong>Reference column</strong></td>
<td>Type in a column of reference that the <strong>tDBOutput</strong> can use to place or replace the new or altered column.</td>
</tr>
<tr>
<td><strong>Use field options</strong></td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td><strong>Enable debug mode</strong></td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
</tbody>
</table>
**tMaxDBOutput**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_UPDATED</strong>: the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_DELETED</strong>: the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_REJECTED</strong>: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a database. It also allows you to create a reject flow using a Row &gt; Rejects link to filter data in error. For an example of tMySqlOutput in use, see Scenario: Retrieving data in error with a Reject link on page 2529.</td>
</tr>
</tbody>
</table>

| Limitation | This component requires installation of its related jar files. |

**Related scenario**

For a related scenario, see:

- Scenario: Inserting a column and altering data using tMysqlOutput on page 2521.
tMaxDBRow

Acts on the actual DB structure or on the data (although without handling data), depending on the nature of the query and the database.

tMaxDBRow is the specific component for this database query. It executes the SQL query stated onto the specified database. The row suffix means the component implements a flow in the job design although it doesn’t provide output.

The SQLBuilder tool helps you write easily your SQL statements.

**tMaxDBRow Standard properties**

These properties are used to configure tMaxDBRow running in the Standard Job framework.

The Standard tMaxDBRow component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Host**

Database server IP address

**Port**

Listening port number of DB server.

**Database**

Name of the database.

**Username and Password**

DB user authentication data.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Table name</th>
<th>Type in the table name.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query type</strong></td>
<td>The query can be <strong>Built-in</strong> for a particular Job, or for commonly used query, it can be stored in the <strong>Repository</strong> to ease the query reuse.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: Fill in manually the query statement or build it graphically using SQLBuilder.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the relevant query stored in the Repository. The <strong>Query</strong> field gets accordingly filled in.</td>
<td></td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Guess Query</strong></td>
<td>Click the <strong>Guess Query</strong> button to generate the query which corresponds to your table schema in the <strong>Query</strong> field.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Propagate QUERY's recordset</th>
<th>Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the <strong>use column</strong> list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use PreparedStatement</strong></td>
<td>Select this check box if you want to query the database using a PreparedStatement. In the <strong>Set PreparedStatement Parameter</strong> table, define the parameters represented by “?” in the SQL instruction of the <strong>Query</strong> field in the <strong>Basic Settings</strong> tab.</td>
</tr>
<tr>
<td><strong>Parameter Index</strong>: Enter the parameter position in the SQL instruction.</td>
<td></td>
</tr>
<tr>
<td><strong>Parameter Type</strong>: Enter the parameter type.</td>
<td></td>
</tr>
</tbody>
</table>
Parameter Value: Enter the parameter value.

Note:
This option is very useful if you need to execute the same query several times. Performance levels are increased

Commit every
Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.

tStat Catcher Statistics
Select this check box to collect log data at the component level.

Global Variables

Global Variables

QUERY: the query statement being processed. This is a Flow variable and it returns a string.

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule
This component offers the flexibility benefit of the DB query and covers all possible SQL queries.

Limitation
This component requires installation of its related jar files.

Related scenario

No scenario is available for this component yet.
tMDMBulkLoad

Uses bulk mode to write XML structured master data into the MDM server.
Note that tMDMBulkLoad does not generate journal entries as part of the bulk load.

**tMDMBulkLoad Standard properties**

These properties are used to configure tMDMBulkLoad running in the Standard Job framework.
The Standard tMDMBulkLoad component belongs to the Talend MDM family.
The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  

- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. Click **Sync columns** to collect the schema from the previous component. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>You create the schema and store it locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td><strong>XML field</strong></td>
<td>Select the name of the column in which you want to write the XML data.</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td>Type in the URL required to access the MDM server.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>Type in the user authentication data for the MDM server.</td>
</tr>
</tbody>
</table>
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

| **Data Model** | Type in the name of the data model against which the data to be written is validated. |
| **Data Container** | Type in the name of the data container where you want to write the master data. |
| **Entity** | Type in the name of the entity that holds the data record(s) you want to write. |
| **Type** | Select **Master** or **Staging** to specify the database on which the action should be performed. |
| **Validate** | Select this check box to validate the data you want to write onto the MDM server against validation rules defined for the current data model. Note that for the PROVISIONING Data Container, validation checks will always be performed on incoming records, regardless of whether or not this check box is selected. For more information on how to set the validation rules, see **Talend Studio User Guide**. **Warning:** If you need faster loading performance, do not select this check box. |
| **Generate ID** | Select this check box to generate an ID number for all of the data written. **Warning:** If you need faster loading performance, do not select this check box. |
| **Insert only** | Select this check box to skip the step of checking whether the data records to be inserted already exist on the MDM server, thus achieving a better performance. However, before using this option, you need to make sure that the data records do not exist in the database. |
| **Commit size** | Type in the row count of each batch to be written onto the MDM server. |
| **Use Transaction** | Select this check box then, in the **Component List**, click an existing connection component which will be used to commit the transaction. |

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the processing metadata at the Job level as well as at each component level. |
## Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |

## Usage

| Usage rule | This component needs always an incoming link to offer XML structured data. If your data offered is not yet in the XML structure, you need use components like **tWriteXMLField** to transform this data into the XML structure. For further information about **tWriteXMLField**, see *tWriteXMLField on page 4047*.  
You can increase the timeout values for a Job using this component to help process a large number of data records. For more information, see advanced execution settings for JVM parameters on Talend Help Center ([https://help.talend.com](https://help.talend.com)).  
If you use a Job with the component **tMDMBulkLoad** to bulk load large volumes of data into MDM, you can tune the bulk load operation by adding a specific JVM argument (for example, `bulkload.concurrent.http.requests=25`) in the **Advanced settings** tab of the Job to limit the maximum number of concurrent requests sent to the MDM server. This avoids consuming all available Tomcat application server connections, which will lead to transaction and deadlock issues. |

## Connections

| Connections | Outgoing links (from this component to another):  
**Row**: Main,  
Incoming links (from one component to this one):  
**Row**: Main  
**Trigger**: Run if, On Component Ok, On Component Error, On Subjob Ok, On Subjob Error  
For further information regarding connections, see *Talend Studio User Guide*. |
Scenario: Loading records into a business entity

This scenario applies only to a Talend solution with MDM.

This scenario describes a Job that loads records into the ProductFamily business entity defined by a specific data model in the MDM hub.

Prerequisites:

- The Product data container: This data container is used to separate the product master data domain from the other master data domains.
- The Product data model: This data model is used to define the attributes, validation rules, user access rights and relationships of the entities of interest. Thus it defines the attributes of the ProductFamily business entity.
- The ProductFamily business entity: This business entity contains Id, Name, both defined by the Product data model.

For further information about how to create a data container, a data model, and a business entity along with its attributes, see the MDM part of your Talend Studio MDM Studio User Guide.

The Job in this scenario uses three components.

- tFixedFlowInput: This component generates the records to be loaded into the ProductFamily business entity. In a real-life project, your records to be loaded are often voluminous and stored in a specific file. However, to simplify the replication of this scenario, this Job uses tFixedFlowInput to generate four sample records.
- tWriteXMLField: This component transforms the incoming data into XML structure.
- tMDMBulkLoad: This component writes the incoming data into the ProductFamily business entity in bulk mode, generating ID value for each of the record data.

Dropping and linking components

Procedure

1. Drop tFixedFlowInput, tWriteXMLField and tMDMBulkLoad onto the design workspace.
2. Connect tFixedFlowInput to tWriteXMLField using the Main link.
3. Do the same to connect tWriteXMLField to tMDMBulkLoad.

Configuring the components

Generating the data records to be loaded into a business entity

Procedure

1. Double click tFixedFlowInput to open its Basic settings view.
2. Click the [...] button next to Edit schema to open the schema editor.

3. In the schema editor, click the [+ ] button to add one row.
4. Name the new column, family in this example.
5. Click OK to close the schema editor.
6. In the Mode area of the Basic settings view, select the Use Inline Table option.
7. Click the [+] button four times to add four rows in the table.
8. In the inline table, click each of the added rows and then enter their names between quotes: Shirts, Hats, Pets, and Mugs.

Transforming the incoming data into XML structure

Procedure
1. Double-click tWriteXMLField to open its Basic settings view.
2. Click the [...] button next to the **Edit schema** field to open the schema editor and then add a row by clicking the + button.

3. Click the newly added row to the right view of the schema editor and enter the name of the output column where you want to write the XML content. It is `xmlRecord` in this example.

4. Click OK to validate this output schema and close the schema editor.

   In the dialog box that pops up, click OK to propagate this schema to the following component.

5. In the **Basic settings** view, click the [...] button next to **Configure XML Tree** to open the dialog box where you can create the XML structure.
6. In the **Link Target** area, click *rootTag* and rename it to *ProductFamily*, which is the name of the business entity used in this scenario.

7. In the **Linker source** area, drop *family* to *ProductFamily* in the **Link target** area.
   
   A dialog box pops up, asking you to select one operation.
   
   Select *Create as sub-element of target node* to create a sub-element of the *ProductFamily* node.
   
   Then, the *family* element appears under the *ProductFamily* node.
   
   Right-click the *Name* node and select from the contextual menu **Set As Loop Element**.

8. In the **Link target** area, click the *family* node and rename it to *Name*, which is one of the attributes of the *ProductFamily* business entity.
   
   Click **OK** to validate the XML structure you defined.

### Writing the incoming data into a business entity

#### Procedure

1. Double-click **tMDMBulkLoad** to open its **Basic settings** view.

   ![tMDMBulkLoad Basic settings](image)

   2. Select *xmlRecord* from the **XML Field** drop-down list.

   3. In the **URL** field, enter the bulk loader URL between quotes. For example, `http://localhost:8180/talendmdm/services/bulkload`.

   4. In the **Username** and **Password** fields, enter your login and password to connect to the MDM server.

   5. In the **Data Model** and the **Data Container** fields, enter the names corresponding to the data model and the data container you need to use. Both are *Product* for this scenario.

   In the **Entity** field, enter the name of the business entity into which you want to load the records. In this example, enter *ProductFamily*.

   6. Select the **Generate ID** check box in order to generate ID values for the records to be loaded.

   7. In the **Commit size** field, type in the batch size to be written into the MDM hub in bulk mode.

### Saving and executing the Job

#### Procedure

1. Press **Ctrl+S** to save your Job.

2. Execute the Job by pressing **F6** or clicking **Run** on the **Run** tab.

   Log into your **Talend MDM Web UI** to check the newly added records for the *ProductFamily* business entity.
<table>
<thead>
<tr>
<th>Id (sequence)</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Shirts</td>
</tr>
<tr>
<td>8</td>
<td>Hats</td>
</tr>
<tr>
<td>9</td>
<td>Pets</td>
</tr>
<tr>
<td>10</td>
<td>Mugs</td>
</tr>
</tbody>
</table>

Product Family

Shirts
tMDMClose

Terminates an open MDM server connection after the execution of the proceeding subjob.

**tMDMClose Standard properties**

These properties are used to configure tMDMClose running in the Standard Job framework.

The Standard tMDMClose component belongs to the Talend MDM family.

The component in this framework is generally available.

**Basic settings**

| Component List | Select the tMDMConnection component from the list if more than one connection is planned for the current Job. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

**Usage**

| Usage rule | This component is to be used along with the tMDMConnection component. Note that the tMDMCommit and tMDMRollback components also include an option in their Basic settings to explicitly close a connection. |

| Dynamic settings | Click the [*] button to add a row in the table and fill the Code field with a context variable to choose your MDM server connection dynamically from multiple connections planned in your Job. |
Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Related scenario

For a related use case, see **Scenario: Deleting master data from an MDM Hub** on page 2138.
tMDMCommit

Commits all changes to the database made within the scope of a transaction in MDM.

This component is used to control the point in an MDM Job at which any changes made to the database within the scope of an MDM transaction are committed, for example to prevent partial commits if an error occurs.

For more information about MDM transactions, see the article MDM Transactions on Talend Help Center (https://help.talend.com).

tMDMCommit Standard properties

These properties are used to configure tMDMCommit running in the Standard Job framework.

The Standard tMDMCommit component belongs to the Talend MDM family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Component List</th>
<th>Select the tMDMConnection component for which you want the commit action to be performed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close Connection</td>
<td>Select this check box to close the session for this connection to the MDM Server after committing the changes. Note that even if you do not select this check box, the connection can still not be used in a subsequent subjob unless the Autocommit mode has been enabled.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

For further information about variables, see Talend Studio User Guide.
Usage

| Usage rule | This component is to be used along with the tMDM Connection, tMDMRollback, tMDMSP, tMDMViewSearch, tMDMInput, tMDMDelete, tMDMRouteRecord, tMDMOutput, and tMDMClose components. |

Related scenario

For a related use case, see Scenario: Deleting master data from an MDM Hub on page 2138.
tMDMConnection

Opens an MDM server connection for convenient reuse in the current Job or transaction.

**tMDMConnection Standard properties**

These properties are used to configure tMDMConnection running in the Standard Job framework.
The Standard tMDMConnection component belongs to the Talend MDM family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDM version</td>
<td>By default, Server 6.0 is selected. Although it is recommended to migrate existing Jobs for this new version, the Server 5.6 option is available to ease the process of the migration of your Jobs so as to keep them working without modification with a 6.0 server. To do so, an option on the server must be enabled to accept and translate requests from such Jobs.</td>
</tr>
<tr>
<td>URL</td>
<td>Type in the URL required to access the MDM server.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Type in the user authentication data for the MDM server. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Auto Commit</td>
<td>Select this check box to commit any changes to the database automatically upon the transaction. With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component. In the MDM context, with the Auto-Commit function enabled, queries sent to the MDM Server will be committed once they are executed. For this reason, if you need more precision in managing your transactions in a Job, it is recommended to use the component tMDMCommit.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>
Global Variables

**Global Variables**

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide.* |

Usage

**Usage rule**

This component is to be used along with the tMDMCommit, tMDMRollback, tMDMSP, tMDMViewSearch, tMDMInput, tMDMDelete, tMDMRouteRecord, tMDMOutput, and tMDMClose components.

Related scenario

For a related use case, see *Scenario: Deleting master data from an MDM Hub* on page 2138.
tMDMDelete

Deletes master data records from specific entities in the MDM Hub.

tMDMDelete Standard properties

These properties are used to configure tMDMDelete running in the Standard Job framework.

The Standard tMDMDelete component belongs to the Talend MDM family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the <strong>Repository Content</strong> window.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Sync columns</strong> to collect the schema from the previous component.</td>
</tr>
<tr>
<td>Built-in</td>
<td><strong>Built-in</strong>: You create the schema and store it locally for this component only. Related topic: see  <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td>Repository</td>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and job designs. Related topic: see  <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box if you want to use a configured <strong>tMDMConnection</strong> component.</td>
</tr>
<tr>
<td>MDM version</td>
<td>By default, <strong>Server 6.0</strong> is selected. Although it is recommended to migrate existing Jobs for this new version, the <strong>Server 5.6</strong> option is available to ease the process of the migration of your Jobs so as to keep them working without modification with a 6.0 server. To do so,</td>
</tr>
</tbody>
</table>
an option on the server must be enabled to accept and translate requests from such Jobs.

<table>
<thead>
<tr>
<th><strong>URL</strong></th>
<th>Type in the URL required to access the MDM server.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Username and Password</strong></td>
<td>Type in the user authentication data for the MDM server. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Entity</strong></td>
<td>Type in the name of the entity that holds the data record(s) you want to delete.</td>
</tr>
<tr>
<td><strong>Data Container</strong></td>
<td>Type in the name of the data container that holds the data record(s) you want to delete.</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Select Master or Staging to specify the database on which the action should be performed.</td>
</tr>
</tbody>
</table>
| **Use multiple conditions** | Select this check box to filter the master data to be deleted, using certain conditions.  
**Xpath**: Enter between quotes the path and the XML node to which you want to apply the condition.  
**Function**: Select the condition to be used from the list.  
**Value**: Enter between inverted commas the value you want to use.  
**Predicate**: Select a predicate if you use more than one condition.  
For more information about the operators and the predicates, see tMDMInput Standard properties on page 2145. |
| **Keys (in sequence order)** | Specify the field(s) (in sequence order) composing the key when the entity have a multiple key. |
| **Fire Delete event** | Select this check box so that a delete event will be generated and written into the UpdateReport, which provides a complete report on all create, update or delete actions on the master data.  
**Note:** The check box is available only if you select Master from the Type list.  
Select this check box so that a delete event will be generated and written into the UpdateReport, which provides a complete report on all create, update or delete actions on the master data.  
- **Source Name**: Define a name between quotes for the source which generates the delete event.  
- **Invoke before deleting process**: Select this check box to enable the execution of a Before Deleting Process to verify the delete event.  
For further information about the UpdateReport and Process types, see Talend Studio User Guide. |
<p>| <strong>Logical delete</strong> | Select this check box to send the master data to the Recycle bin and fill in the Recycle bin path. Once in the Recycle bin, the master data can be permanently deleted |</p>
<table>
<thead>
<tr>
<th><strong>tMDMDelete</strong></th>
<th>or restored. If you leave this check box clear, the master data will be permanently deleted.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Die on error</strong></td>
<td>Select this check box to skip the row in error and complete the process for error-free rows. If needed, you can retrieve the rows in error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

**Global Variables**

| **Global Variables** | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
**NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see **Talend Studio User Guide**. |

**Usage**

| **Usage rule** | If the **Use multiple conditions** check box is selected, **tMDMDelete** can work as a standalone component. Otherwise, it needs an input connection.  
You can increase the timeout values for a Job using this component to help process a large number of data records. For more information, see advanced execution settings for JVM parameters on Talend Help Center ([https://help.talend.com](https://help.talend.com)). |

**Scenario: Deleting master data from an MDM Hub**

This scenario applies only to a Talend solution with MDM.

This scenario describes a six-component Job that deletes the specified data record from the MDM Hub. It also makes use of the **tMDMCommit** and **tMDMRollback** components so that changes to the database are only made on successful completion of the Job.
Dropping and linking the components

Procedure

1. Drop tMDMConnection, tMDMCommit, tMDMClose, tMDMInput, tMDMDelete, and tMDMRollback from the Talend MDM family from the Palette onto the design workspace.
2. Connect tMDMInput to tMDMDelete using a Row > Main link.
3. Connect tMDMConnection to tMDMInput using a Trigger > OnSubjobOk link.
4. Connect tMDMDelete to tMDMCommit, tMDMCommit to tMDMClose and tMDMRollback to tMDMClose using Trigger > OnComponentOk links.
5. Connect tMDMDelete to tMDMRollback using a Trigger > OnComponentError link.

Configuring the MDM server connection

About this task

In this scenario, a tMDMConnection component is used to open an MDM server connection for convenient reuse in the subsequent subjob that performs the data record deletion task.

Procedure

1. Double-click tMDMConnection to display its Basic settings view and define the component properties.
   
   ![tMDMConnection_1](image)

   **Basic settings**

   - **MDM version**: Server 5.0
   - **URL**: "http://localhost:8180/talendmdm/services/soap"
   - **Username**: "userName"
   - **Password**: ************
   - **Auto Commit**: checked

2. In the URL field, enter the MDM server URL, between quotation marks: for example, ‘http://localhost:8180/talendmdm/services/soap’.
3. In the **Username** and **Password** fields, enter your user name and password to connect to the MDM server.

4. Double-click **tMDMCommit** to display its **Basic settings** view and define the component properties.

   ![tMDMCommit Component](image)

   This component commits the changes made to the database on successful completion of the proceeding subjob.

5. From the **Component List** list, select the component for the server connection you want to close if you have configured more than one MDM server connection. In this use case, there is only one MDM server connection open, so simply use the default setting.

6. Deselect the **Close Connection** check box if it is selected. In this example, the **tMDMClose** component closes the connection to the MDM Server.

7. Double-click **tMDMRollback** to display its **Basic settings** view and define the component properties.

   ![tMDMRollback Component](image)

   This component rolls back any changes and returns the database to its previous state if the proceeding subjob fails.

8. From the **Component List** list, select the component for the server connection you want to close if you have configured more than one MDM server connection. In this use case, there is only one MDM server connection open, so simply use the default setting.

9. Deselect the **Close Connection** check box if it is selected. In this example, the **tMDMClose** component closes the connection to the MDM Server.

10. Double-click **tMDMClose** to display its **Basic settings** view and define the component properties.

    The **tMDMClose** component is used to close the connection after the successful execution of the Job. You can also close the connection by selecting the **Close Connection** check box in the **tMDMCommit** and **tMDMRollback** components, but for the purposes of this scenario the **tMDMClose** component is used instead.
11. From the **Component List** list, select the component for the server connection you want to close if you have configured more than one MDM server connection. In this use case, there is only one MDM server connection open, so simply use the default setting.

## Configuring data retrieval

### Procedure

1. Double-click **tMDMInput** to display its **Basic settings** view and define the component properties.

2. From the **Property Type** list, select **Built-in** to complete the fields manually.
   - If you have stored the MDM connection information in the repository metadata, select **Repository** from the list and the fields will be completed automatically.

3. From the **Schema** list, select **Built-in** and click `[...]` next to **Edit schema** to open a dialog box.
   - Here you can define the structure of the master data you want to read in the MDM hub.
4. The master data is collected in a three-column schema of the type String: Id, Name and Price. Click OK to close the dialog box and proceed to the next step.

5. Select the Use an existing connection check box, and from the Component List list that appears, select the component you have configured to open your MDM server connection. In this scenario, only one MDM server connection exists, so simply use the default selection.

6. In the Entity field, enter the name of the business entity that holds the data record(s) you want to read, between quotation marks. Here, we want to access the Product entity.

7. In the Data Container field, enter the name of the data container that holds the master data you want to read, between quotation marks. In this example, we use the Product container.

   **Note:** The Use multiple conditions check box is selected by default.

8. In the Operations table, define the conditions to filter the master data you want to delete as follows:

   a) Click the plus button to add a new line.

   b) In the Xpath column, enter the Xpath and the tag of the XML node on which you want to apply the filter, between quotation marks. This example works with the Product entity, so enter “Product/Name”.

   c) In the Function column, select the function you want to use. In this scenario, use the Contains function.

   d) In the Value column, enter the value of your filter. This example is to filter the master data where the Name contains mug.

9. In the Component view, click Advanced settings to set the advanced parameters.
10. In the **Loop XPath query** field, enter the structure and the name of the XML node on which the loop is to be carried out, between quotation marks.

11. In the **Mapping** table and in the **XPath query** column, enter the name of the XML tag in which you want to collect the master data, next to the corresponding output column name, between quotation marks.

### Configuring data record deletion

#### Procedure

1. In the design workspace, double-click the **tMDMDelete** component to display the **Basic settings** view and set the component properties.

2. From the **Schema** list, select **Built-in** and click the three-dot button next to the **Edit Schema** field to describe the structure of the master data in the MDM hub.
3. Click the plus button to the right to add one column of the type String. In this example, name this column outputXML. Click OK to close the dialog box and proceed to the next step.

4. Select the Use an existing connection check box, and from the Component List list that appears, select the component you have configured to open your MDM server connection. In this scenario, only one MDM server connection exists, so simply use the default selection.

5. In the Entity field, enter the name of the business entity that holds the master data you want to delete, the Product entity in this example.

6. In the Data Container, enter the name of the data container that holds the data to be deleted, Product in this example.

7. In the Keys table, click the plus button to add a new line. In the Keys column, select the column that holds the key of the Product entity. Here, the key of the Product entity is set on the Id field.

   Note:
   If the entity has multiple keys, add as many line as required for the keys and select them in sequential order.

8. If needed, select the Fire Delete event check box, so that the delete event will be generated and written into UpdateReport.

9. Select the Logical delete check box if you do not want to delete the master data permanently. This will send the deleted data to the Recycle bin. Once in the Recycle bin, the master data can be restored or permanently deleted. If you leave this check box clear, the master data will be permanently deleted.

Saving and executing the Job

Procedure

1. Press Ctrl+S to save your Job to ensure that all the parameters you have configured take effect.
2. Press F6 to execute your Job.

Results

The master data with the Name containing "mug" have been deleted and sent to MDM Recycle bin.
tMDMInput

Reads data in an MDM Hub and thus makes it possible to process this data.

**tMDMInput Standard properties**

These properties are used to configure tMDMInput running in the Standard Job framework.

The Standard tMDMInput component belongs to the Talend MDM family.

The component in this framework is generally available.

**Basic Settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in:</strong> No property data stored centrally</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> Select the repository file where properties are stored. The fields that follow are completed automatically using the fetched data</td>
</tr>
</tbody>
</table>

**Schema and Edit Schema**

A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Use an existing connection**

Select this check box if you want to use a configured tMDMConnection component.

**MDM version**

By default, **Server 6.0** is selected. Although it is recommended to migrate existing jobs for this new version.
version, the **Server 5.6** option is available to ease the process of the migration of your Jobs so as to keep them working without modification with a 6.0 server. To do so, an option on the server must be enabled to accept and translate requests from such Jobs.

<table>
<thead>
<tr>
<th><strong>URL</strong></th>
<th>Type in the URL to access the MDM server.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Username and Password</strong></td>
<td>Type in user authentication data for the MDM server. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Entity</strong></td>
<td>Type in the name of the business entity that holds the data you want to read.</td>
</tr>
<tr>
<td><strong>Data Container</strong></td>
<td>Type in the name of the data container that holds the data you want to read.</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Select <strong>Master</strong> or <strong>Staging</strong> to specify the database on which the action should be performed.</td>
</tr>
</tbody>
</table>
| **Use multiple conditions** | Select this check box to filter the data using certain conditions. **Xpath**: Enter between quotes the path and the XML node to which you want to apply the condition. **Function**: Select the condition to be used from the list. Before using the conditions, bear in mind the following:

- Depending on the type of field the **Xpath** points to, only certain operators apply. For example, if the field is a boolean, only the **Equal** or **Not Equal** operators are appropriate.
- The operator **Not Equal** does not support multi-occurrence fields or complex type fields.

The following operators are available:

- **Contains**: Returns a result which contains the word or words entered. Note that full text search does not support special characters, for example, @, #, $.
- **Contains the sentence**: Returns one or more results which contain the sentence entered.
- **Joins With**: This operator is reserved for future use.
- **Starts With**: Returns a result which begins with the string entered.
- **Equal**: Returns a result which matches the value entered.
- **Not Equal**: Returns a result of any value other than the null value and the value entered.
- **is greater than**: Returns a result which is greater than the numerical value entered. Applies to number fields only. |
• **is greater or equal**: Returns a result which is greater than or equal to the numerical value entered. Applies to number fields only.

• **is lower than**: Returns a result which is less than the numerical value entered. Applies to number fields only.

• **is lower or equal**: Returns a result which is less than or equal to the numerical value entered. Applies to number fields only.

• **whole content contains**: Performs a plain text search using the specified XPath field in the selected data container. If you enter an empty string "" in the Xpath field and select whole content contains from the Function list, searches will be performed in all the fields of all entities in the selected data container.

• **is empty or null**: Returns an empty field or a null value.

**Value**: Enter between inverted commas the value you want to use. Note that if the value contains XML special characters such as /, you must also enter the value in single quotes ("'ABC/XYZ'") or the value will be considered as an XPath.

**Predicate**: Select a predicate if you use more than one condition.

The following predicates are available:

• **Default**: Interpreted as an and.

• **or**: One of the conditions applies.

• **and**: Both or all of the conditions apply.

The other predicates are reserved for future use and may be subject to unpredictable behavior.

If you clear this check box, you have the option of selecting particular IDs to be displayed in the ID value column of the IDS table.

**Note:**
If you clear the **Use multiple conditions** check box, the **Batch Size** option in the **Advanced Settings** tab will no longer be available.

<table>
<thead>
<tr>
<th>Skip Rows</th>
<th>Enter the number of lines to be ignored.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Rows</td>
<td>Maximum number of rows to be processed. If Limit = 0, no row is read or processed.</td>
</tr>
<tr>
<td>Die on error</td>
<td>Select this check box to skip the row in error and complete the process for error-free rows. If needed, you can retrieve the rows in error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>
Advanced settings

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Number of lines in each processed batch.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note:

This option is not displayed if you have cleared the **Use multiple conditions** check box in the **Basic settings** view.

<table>
<thead>
<tr>
<th>Loop XPath query</th>
<th>The XML structure node on which the loop is based.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Mapping</th>
<th>Column: reflects the schema as defined in the Edit schema editor.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XPath query: Type in the name of the fields to extract from the input XML structure.</td>
</tr>
<tr>
<td></td>
<td>Get Nodes: Select this check box to retrieve the Xml node together with the data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <a href="https://help.talend.com">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Use this component as a start component. It needs an output flow.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If an MDM entity has a large number of foreign key fields, it may take you a long time to query data records that pertain to the entity using the tMDMInput component.</td>
</tr>
<tr>
<td></td>
<td>For more information about the workaround, see the article How to read data effectively that pertains to an MDM entity with many foreign key fields on <a href="https://help.talend.com">Talend Help Center</a>.</td>
</tr>
<tr>
<td></td>
<td>You can increase the timeout values for a Job using this component to help process a large number of data.</td>
</tr>
</tbody>
</table>
Reading master data from an MDM hub

This scenario applies only to a Talend solution with MDM.

This scenario describes a two-component Job that fetches master data from an MDM server, and displays the data in the log console.

Prerequisites:

- Make sure the MDM server is up and running.
- You have imported the MDM demo project and loaded the sample data into the data container Product by running the Job MDM_LoadAll.

Creating a Job to read master data from MDM

Procedure

1. From the Palette, drop tMDMInput and tLogRow onto the design workspace.
2. Link the two components together using a Row > Main connection.

Configuring basic settings of tMDMInput to read master data from MDM

Procedure

1. Double-click tMDMInput to open the Basic settings view.
2. In the **Property Type** list, select **Built-In**.

3. In the **Schema** list, select **Built-In** and click the [...] button next to **Edit schema** to open a dialog box and define the structure of the master data you want to read from the MDM server.

   In this example, three columns are defined to fetch three elements from the *Product* entity: *Name*, *Price*, and *Colors*.

![tMDMInput_1](image)

   ![tMDMInput_1](image)

4. After you have defined the schema, click **OK** to close this dialog box, and then click **Yes** in the **[Propagate]** dialog box to propagate the schema changes to **tLogRow**.

5. Enter the user name and password for accessing the MDM server.

6. In the **Entity** field, enter *Product* between quotes.

7. In the **Data Container** field, enter *Product* between quotes.

8. Select **Master** from the **Type** list.

9. Define the query conditions in the **Operations** area.

   In this example, we want to query the product data records whose names include *Shirt*.

   a) Click the [+ ] button to add a row.

   b) Enter *Product/Name* between quotes in the **Xpath** field.

   Apart from elements defined in entities, you can query metadata elements which are also known as built-in elements. To query metadata elements from records in master database, you need to follow the format of **metadata:<timestamp|task_id>** when defining the path expression to select the XML node to run the query on.

   c) Select **Contains** from the **Function** list.

   d) Enter *Shirt* between quotes in the **Value** field.

**Configuring advanced settings of tMDMInput to read master data from MDM**

**Procedure**

1. In the **Component** view, click the **Advanced settings** tab.
2. In the Loop XPath query field, enter /Product between quotes on which the loop is based.

3. In the XPath query column of the Mapping table, enter the name of the XML node from which you want to collect the master data, next to the corresponding output column name.

4. Select the Get Nodes check box for the Colors row to retrieve the XML node together with its data.

**Configuring the data display mode and executing the Job**

**Procedure**

1. Double-click the tLogRow component to display its Basic settings view.
2. In the Mode area, select Table (print values in cells of a table) for better readability of the result.
3. Save the Job and press F6 to run it.

The Product data records whose names include "Shirt" are displayed on the console with the values of three specified columns.

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Talend Dog T-Shirt</td>
<td>16.99</td>
<td>&lt;Colors&gt;&lt;Color&gt;White&lt;/Color&gt;&lt;Color&gt;Light Pink&lt;/Color&gt;&lt;Colors&gt;</td>
</tr>
<tr>
<td>Talend Golf Shirt</td>
<td>16.99</td>
<td>&lt;Colors&gt;&lt;Color&gt;White&lt;/Color&gt;&lt;Colors&gt;</td>
</tr>
<tr>
<td>Talend Fitted T-Shirt</td>
<td>16.99</td>
<td>&lt;Colors&gt;&lt;Color&gt;White&lt;/Color&gt;&lt;Color&gt;Light Blue&lt;/Color&gt;&lt;Colors&gt;</td>
</tr>
</tbody>
</table>
```
tMDMOutput

Writes data into or removes data from the MDM server.

tMDMOutput receives data from the preceding component, generates an XML document, and then writes data in an MDM Hub using an output field.

**tMDMOutput Standard properties**

These properties are used to configure tMDMOutput running in the Standard Job framework.

The Standard tMDMOutput component belongs to the Talend MDM family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file where the properties are stored. The fields which follow are filled in automatically using the fetched data.</td>
</tr>
</tbody>
</table>

**Input Schema and Edit schema**

An input schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.

Click **Sync columns** to collect the schema from the previous component.

**Built-in**: You create the schema and store it locally for this component only. Related topic: see **Talend Studio User Guide**.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various
Build the document | Select this check box if you want to build the document from a flat schema. If this is the case, double-click the component and map your schema in the dialog box that opens.

If the check box is not selected, you must select the column in your schema that contains the document from the Predefined XML document list.

Result of the XML serialization | Lists the name of the XML output column that will hold the XML data.

Use an existing connection | Select this check box if you want to use a configured tMDMConnection component.

MDM version | By default, Server 6.0 is selected. Although it is recommended to migrate existing Jobs for this new version, the Server 5.6 option is available to ease the process of the migration of your Jobs so as to keep them working without modification with a 6.0 server. To do so, an option on the server must be enabled to accept and translate requests from such Jobs.

URL | Type in the URL of the MDM server.

Username and Password | Type in the user authentication data for the MDM server.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

Note:
Ensure that the user has been assigned a role in Talend MDM enabling him or her to connect through a Job or any other web service call. For further information, see Talend Studio User Guide.

Data Model | Type in the name of the data model against which the data to be written is validated.

Data Container | Type in the name of the data container where you want to write the master data.

Note:
This data container must already exist.

Type | Select Master or Staging to specify the database on which the action should be performed.

Return Keys | Columns corresponding to IDs in order: in sequential order, set the output columns that will store the return key values (primary keys) of the item(s) that will be created.
| **Is Update** | Select this check box to update the modified fields. If you leave this check box unchecked, all fields will be replaced by the modified ones. |
| **Fire Create/Update event** | Select this check box to add the actions carried out to a modification report.  
*Source Name*: Between quotes, enter the name of the application to be used to carry out the modifications.  
*Enable verification by "before saving" process*: Select this check box to verify the commit that has been just added; prior to saving. |
| **Use partial update** | Select this check box if you need to update multi-occurrence elements (attributes) of an existing item (entity) based on the content of a source XML stream. Once selected, you need to set the parameters presented below:  
- **Pivot**: type in the xpath to the multi-occurrences sub-element where data need to be added, replaced or deleted in the item of interest.  
For example, if you need to add a child sub-element to the below existing item:  
```
<Person>  
  <Id>1</Id> <!-- record key is mandatory -->  
  <Children>  
    <Child>[1234]</Child> <!-- FK to a Person Entity -->  
  </Children>  
</Person>
```
then the Xpath you enter in this **Pivot** field must read as follows: `/Person/Children/Child` where the **Overwrite** check box is cleared.  
And, if you need to replace a child sub-element in an existing item:  
```
<Person>  
  <Id>1</Id>  
  <Addresses>  
    <Address>  
      <Type>office</Type>  
      (...address elements are here....)  
    </Address>  
    <Address>  
      <Type>home</Type>  
      (...address elements are here....)  
    </Address>  
  </Addresses>  
</Person>
```
then the Xpath you enter in this **Pivot** field must read as follows: `/Person/Addresses/Address` where the **Overwrite** check box is selected, and the **Key** field is set to `/Type`. |
In such an example, assuming the item in MDM only has an office address, the office address will be replaced, and the home address will be added.

- **Overwrite**: select this check box if you need to replace or update the original sub-elements with the input sub-elements. Leave unselected if you want to add a sub-element.

- **Key**: type in the xpath relative to the pivot that will help match a sub-element of the source XML with a sub-element of the item. If a key is not supplied, all sub-elements of an item with an XPath matching that of the sub-element of the source XML will be replaced. If more than one sub-element matches the key, MDM will update the first one it encounters. If no sub-elements match the key, it is added at the end of the collection.

- **Position**: type in a number to indicate the position after which the new elements (those that do not match the key) will be added. If you do not provide a value in this field, the new element will be added at the end.

- **Delete**: select this check box if you need to remove one or more sub-elements from the original sub-elements.

For example, if you need to remove two houses from the existing item below:

```xml
<Person>
  <Id>1</Id>
  <Name>p1</Name>
  <Houses>
    <House>[1]</House>
    <House>[3]</House>
  </Houses>
  <Children>
    <Child>
      <Name>k1</Name>
      <Age>1</Age>
      <Habits>
        <Habit>Basketball</Habit>
        <Habit>Football</Habit>
        <Habit>Tennis</Habit>
        <Habit>Boxing</Habit>
      </Habits>
    </Child>
    <Child>
      <Name>k2</Name>
      <Age>2</Age>
    </Child>
  </Children>
</Person>
```

then the Xpath you enter in this **Pivot** field must read as follows: `/Person/Houses/House` where the **Delete** check box is selected, and the **Key** field is set to . or empty. Moreover, you need to provide the source XML stream as follows:

```xml
<Person>
  <Id>1</Id>
  <Houses>
    <House>[1]</House>
  </Houses>
</Person>
```
In this case, the House [1] and House [2] will be deleted.

For more examples of the partial update operations, see Examples of partial update operations using tMDMOutput on page 2157.

| **Die on error** | Select this check box to skip the row in error and complete the process for error-free rows. If needed, you can retrieve the rows in error via a Row > Rejects link. |

### Advanced settings

| **Extended Output** | Select this check box to commit master data in batches. You can specify the number of lines per batch in the Rows to commit field. |
| **Configure Xml Tree** | Opens the interface which helps create the XML structure of the master data you want to write. |
| **Group by** | Select the column to be used to regroup the master data. |
| **Create empty element if needed** | This check box is selected by default. If the content of the interface’s Related Column which enables creation of the XML structure is null, or if no column is associated with the XML node, this option creates an opening and closing tag at the required places. |
| **Advanced separator (for number)** | Select this check box to modify the number of separators used by default. |
| - **Thousands separator:** enter between inverted commas the separator for thousands. |
| - **Decimal separator:** enter between inverted commas the decimal separator. |
| **Generation mode** | Select the appropriate generation mode according to your memory availability. The available modes are: |
| - **Slow and memory-consuming (Dom4j)** |
| ![Note](https://via.placeholder.com/15) **Note:** This option allows you to use dom4j to process the XML files of high complexity. |
| - **Fast with low memory consumption** |
| **Encoding** | Select the encoding type from the list or else select Custom and define it manually. This is an obligatory field for the manipulation of data on the server. |
| **tStatCatcher Statistics** | Select this check box to gather the processing metadata at the Job level as well as at each component level. |
Global Variables

| Global Variables |  
|------------------|---|
| **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
| **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
| **NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer.  
| A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
| To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
| For further information about variables, see *Talend Studio User Guide*. |

Usage

| Usage rule |  
|------------|---|
| Use this component to write a data record and separate the fields using a specific separator.  
| You can increase the timeout values for a Job using this component to help process a large number of data records. For more information, see advanced execution settings for JVM parameters on Talend Help Center ([https://help.talend.com](https://help.talend.com)). |

Examples of partial update operations using tMDMOutput

The section will demonstrate how to set the parameters in the component tMDMOutput to perform the partial update operations with several examples.

Partial update with Overwrite enabled or disabled

The following two examples explain how to add some elements to an existing item (entity) with multi-occurrence elements (attributes) and then update the newly added elements based on the content of a source XML stream.

Given an existing item as follows:

```xml
<Person>
  <Id>1</Id>
  <Name>p1</Name>
  <Houses>
    <House>[1]</House>
    <House>[3]</House>
  </Houses>
  <Children>
    <Child>
```

```xml
```
Example 1: If you want to add two Child items to the existing item, then the XPath you enter in the Pivot field must read as follows: /Person/Children/Child where the Overwrite check box is cleared, and the Key field is set to /Name. Note that the Child element is of complex type, and needs to be identified with the Key field. Moreover, you need to provide the source XML stream as follows:

```
<Person>
  <Id>1</Id>
  <Children>
    <Child>
      <Name>k2</Name>
    </Child>
    <Child>
      <Name>k3</Name>
    </Child>
  </Children>
</Person>
```

In this case, the two child items Child [k2] and Child [k3] will be added, and you will get the following result:

```
<Person>
  <Id>1</Id>
  <Name>p1</Name>
  <Houses>
    <House>[1]</House>
    <House>[3]</House>
  </Houses>
  <Children>
    <Child>
      <Name>k1</Name>
      <Age>1</Age>
      <Habits>
        <Habit>Basketball</Habit>
        <Habit>Football</Habit>
        <Habit>Tennis</Habit>
        <Habit>Boxing</Habit>
      </Habits>
    </Child>
    <Child>
      <Name>k2</Name>
    </Child>
    <Child>
      <Name>k3</Name>
    </Child>
  </Children>
</Person>
```

Example 2: If you want to change the names of the two child items Child [k2] and Child [k3], then the XPath you enter in the Pivot field must read as follows: /Person/Children/Child where the
Overwrite check box is selected, and the Key field is set to /Name. Moreover, you need to provide the source XML stream as follows:

```xml
<Person>
  <Id>1</Id>
  <Children>
    <Child>
      <Name>Tina</Name>
    </Child>
    <Child>
      <Name>Tommy</Name>
    </Child>
  </Children>
</Person>
```

In this case, the names of the two child items Child [k2] and Child [k3] will be updated, and you will get the following result:

```xml
<Person>
  <Id>1</Id>
  <Name>p1</Name>
  <Houses>
    <House>[1]</House>
    <House>[3]</House>
  </Houses>
  <Children>
    <Child>
      <Name>k1</Name>
      <Age>1</Age>
      <Habits>
        <Habit>Basketball</Habit>
        <Habit>Football</Habit>
        <Habit>Tennis</Habit>
        <Habit>Boxing</Habit>
      </Habits>
    </Child>
    <Child>
      <Name>Tina</Name>
    </Child>
    <Child>
      <Name>Tommy</Name>
    </Child>
  </Children>
</Person>
```

**Partial update with Delete enabled**

The following examples show how to remove some elements from an existing item (entity) with multi-occurrence elements (attributes) based on the content of a source XML stream.

Given an existing item as follows:

```xml
<Person>
  <Id>1</Id>
  <Name>p1</Name>
  <Houses>
    <House>[1]</House>
    <House>[3]</House>
  </Houses>
  <Children>
```
<Child>
  <Name>k1</Name>
  <Age>1</Age>
  <Habits>
    <Habit>Basketball</Habit>
    <Habit>Football</Habit>
    <Habit>Tennis</Habit>
    <Habit>Boxing</Habit>
  </Habits>
</Child>

<Child>
  <Name>k2</Name>
  <Age>2</Age>
</Child>

<Child>
  <Name>k3</Name>
  <Age>3</Age>
</Child>

</Children>

</Habits>

</Pets>

</Pets>

</Pets>

</Pets>

</Pets>

</Person>

Example 1: If you want to remove two child items from the existing item, then the Xpath you enter in the Pivot field must read as follows: /Person/Children/Child where the Delete check box is selected, and the Key field is set to /Name. Note that the Child element is of a complex type, and needs to be identified with the Key field. Moreover, you need to provide the source XML stream as follows:

<Person>
  <Id>1</Id>
  <Children>
    <Child>
      <Name>k2</Name>
    </Child>
    <Child>
      <Name>k3</Name>
    </Child>
  </Children>
</Person>

In this case, the two items Child [k2] and Child [k3] will be removed, and you will get the following result:

<Person>
  <Id>1</Id>
Example 2: If you want to remove some habits for the Person element, then the Xpath you enter in the Pivot field must read as follows: /Person/Habits/Habit where the Delete check box is selected, and the Key field is set to. or empty. Moreover, you need to provide the source XML stream as follows:

```xml
<Person>
  <Id>1</Id>
  <Habits>
    <Habit>Basketball</Habit>
    <Habit>Football</Habit>
    <Habit>Boxing</Habit>
  </Habits>
</Person>
```

In this case, the three habit items Habit [Basketball], Habit [Football] and Habit [Boxing] will be removed, and you will get the following result:

```xml
<Person>
  <Id>1</Id>
  <Name>p1</Name>
  <Habits>
    <Habit>Tennis</Habit>
  </Habits>
</Person>
```

Example 3: If you want to remove some habits for the Child [k1] element, then the Xpath you enter in the Pivot field must read as follows: /Person/Children/Child[1]/Habits/Habit where the Delete check box is selected, and the Key field is set to. or empty. Note that Child[1] is used in the Pivot field to identify the habits of which child will be removed. Moreover, you need to provide the source XML stream as follows:

```xml
<Person>
  <Id>1</Id>
  <Children>
    <Child>
      <Habits>
        <Habit>Basketball</Habit>
        <Habit>Football</Habit>
      </Habits>
    </Child>
  </Children>
</Person>
```
In this case, the two items Habit [Basketball] and Habit [Football] will be removed, and you will get the following result:

```xml
<Person>
  <Id>1</Id>
  <Name>p1</Name>
  ...
  <Children>
    <Child>
      <Name>k1</Name>
      <Age>1</Age>
      <Habits>
        <Habit>Tennis</Habit>
        <Habit>Boxing</Habit>
      </Habits>
    </Child>
    <Child>
      <Name>k2</Name>
      <Age>2</Age>
    </Child>
    <Child>
      <Name>k3</Name>
      <Age>3</Age>
    </Child>
  </Children>
  ...
</Person>
```

Example 4: If you want to remove some pets for the Person element, then the Xpath you enter in the Pivot field must read as follows: /Person/Pets where the Delete check box is selected, and the Key field is set to /Pet. Moreover, you need to provide the source XML stream as follows:

```xml
<Person>
  <Id>1</Id>
  <Pets>
    <Pet>Cat</Pet>
  </Pets>
  <Pets>
    <Pet>Pig</Pet>
  </Pets>
</Person>
```

In this case, the two items Pet [Cat] and Pet [Pig] will be removed, and you will get the following result:

```xml
<Person>
  <Id>1</Id>
  <Name>p1</Name>
  ...
  <Pets>
    <Pet>Dog</Pet>
  </Pets>
  <Pets>
    <Pet>Cow</Pet>
  </Pets>
</Person>
```

**Scenario: Writing master data in an MDM hub**

This scenario applies only to a Talend solution with MDM.
This scenario describes a two-component Job that generates a data record, transforms it into XML and loads it into the defined business entity in the MDM server.

In this example, we want to load a new agency in the *Agency* business entity. This new agency has an id, a name and three offices located in different cities.

For more information about entities, see *Talend Studio User Guide*.

---

**Dropping and linking the components**

**Procedure**

1. From the *Palette*, drop *tFixedFlowInput* and *tMDMOutput* onto the design workspace.
2. Connect the components using a *Row Main* link.

**Configuring the components**

**Preparing the data to be loaded into the MDM server**

**Procedure**

1. Double-click *tFixedFlowInput* to view its *Basic settings* in the *Component* tab.
2. In the Schema list, select **Built-In** and then click the three-dot button next to **Edit schema** to open a dialog box in which you can define the structure of the master data you want to write on the MDM server.

3. Click the [+] button and add five columns of the type **String**. In this example, name the columns *Id*, *Name*, *Office_R_and_D*, *Office_Sales*, and *Office_Services*.

4. Click **OK** to validate your changes.

5. In the **Number of rows** field, enter the number of rows you want to generate.

6. In the **Mode** area, select the **Use Single Table** option.

7. In the **Value** fields, enter between quotes the values which correspond to each of the schema columns.

**Basic settings of tMDMOutput**

**Procedure**

1. In the design workspace, click **tMDMOutput** to open its **Basic settings** view.
2. In the Input Schema list, select Built-In and then click the [...] button next to the Edit Schema field to define the structure of the master data you want to load into the MDM server. The tMDMOutput component basically generates an XML document, writes it in an output field, and then sends it to the MDM server.

3. Click OK to proceed to the next step. The Result of the XML serialization list in the Basic settings view is automatically filled in with the output xml column.

4. In the URL field, enter the URL to access the MDM server.

5. In the Username and Password fields, enter the authentication information required to connect to the MDM server.

6. In the Data Model field, enter between quotes the name of the data model against which you want to validate the master data you want to write.

7. In the Data Container, enter between quotes the name of the data container into which you want to write the master data.
8. Select the Is Update checkbox if you only want to update some fields rather than the entire data record.

**Advanced settings of tMDMOutput**

**Procedure**

1. In the Component view, click Advanced settings to set the advanced parameters for the tMDMOutput component.

2. Select the Extended Output check box if you want to commit master data in batches, and specify the number of lines per batch in the Rows to commit field.

   Click the [...] next to Configure XML Tree to open the tMDMOutput editor.

   Alternatively, double-click tMDMOutput to open the editor.

3. In the Link target area to the right, click in the XML Tree field and then replace rootTag with the name of the business entity into which you want to insert the data record, **Agency** in this example.
4. In the Linker source area, select the two schema columns *Id* and *Name* and drop them on the Agency node respectively.

The [Selection] dialog box is displayed.

Select the Create as sub-element of target node option so that the two columns are linked to the two XML sub-elements of the Agency node.

5. Right-click the root node Agency and then select Add Sub-element.

In the dialog box that pops up, enter a name for the new sub-element, *Offices* in this example. Repeat the same procedure to create three new sub-elements *Office* for the Offices node which corresponds to the multi-occurrence element Offices of the business entity Agency.

6. In the Linker source area, select the three schema columns *Office_R_and_D*, *Office_Sales* and *Office_Services* and drop them on the three new Office nodes respectively.

The [Selection] dialog box is displayed.

Select the Create as sub-element of target node option so that the three columns are linked to the three XML sub-elements of the Offices node.

7. Click Ok to proceed to the next step.

8. Right-click the element in the Link Target area you want to set as a loop element and select Set As Loop Element from the contextual menu.

In this example, *Id* is the iterating object.

9. Click OK to validate your changes and close the dialog box.

**Saving and executing the Job**

**Procedure**

1. Press Ctrl+S to save your Job.

2. Execute the Job by pressing F6 or clicking Run on the Run tab.

   The new data record is inserted in the Agency business entity in the DStar data container on the MDM server. This data record holds, as you defined in the schema, the agency id, the agency name and the agency offices located in three cities.
Removing master data partially from the MDM hub

This scenario applies only to a Talend solution with MDM.

The scenario describes how to partially remove the master data which has been written into the MDM server in the scenario Scenario: Writing master data in an MDM hub on page 2162.

In this example, one agency office will be removed from the Agency business entity. The agency now has an id, a name and three offices located in different cities.
Dropping and linking the components

Procedure
1. From the Palette, drop **tFixedFlowInput** and **tMDMOutput** onto the design workspace.
2. Connect the components using a **Row Main** link.

Configuring the components

Specifying the data to be removed from the MDM server

Procedure
1. Double-click **tFixedFlowInput** to view its **Basic settings** in the **Component** tab.
   2. In the **Schema** list, select **Built-In** and then click the three-dot button next to **Edit schema** to open a dialog box in which you can define the structure of the master data to be used for partially removing the master data on the MDM server.
3. Click the [+\(\)] button and add three columns of the type **String**.  
   In this example, name the columns *Id*, *Name*, and *Remove.Office*.

4. Click **OK** to save your changes.

5. In the **Number of rows** field, enter the number of rows you want to generate.

6. In the **Mode** area, select the **Use Single Table** option.

7. In the **Value** fields, enter values which correspond to each of the schema columns.  
   In this example, the office in *Paris* will be removed.

**Basic settings of tMDMOutput**

**Procedure**

1. In the design workspace, click **tMDMOutput** to open its **Basic settings** view.
2. In the **Input Schema** list, select **Built-In** and then click **Sync columns**. After receiving data from the previous component, the **tMDMOutput** component basically generates an XML document, writes it in an output field, and then sends it to the MDM server.

3. Click **OK** to proceed to the next step.
The Result of the XML serialization list in the Basic settings view is automatically filled in with the output xml column.

4. In the URL field, enter the URL to access the MDM server.

5. In the Username and Password fields, enter the authentication information required to connect to the MDM server.

6. In the Data Model field, enter between quotes the name of the data model against which you want to validate the master data you want to write.

7. In the Data Container, enter between quotes the name of the data container into which you want to write the master data.

8. In the Partial Update area, select the Use Partial Update check box.
   In the Source Name filed that pops up with your selection, enter the name to be used in the modification report.

9. In the Pivot field, enter the xpath to the multi-occurrence sub-element where data need to be removed.
   In this example, enter "Agency/Offices/Office".

10. Select the Delete check box, and then enter "." in the Key field.

Advanced settings of tMDMOutput

Procedure

1. In the Component view, click Advanced settings to set the advanced parameters for the tMDMOutput component.

2. Click the [...] next to Configure XML Tree to open the tMDMOutput editor.
   Alternatively, double-click tMDMOutput to open the editor.

3. In the Link target area to the right, click in the XML Tree field and then replace rootTag with the name of the business entity in which you want to remove data partially, Agency in this example.
4. In the Linker source area, select the two schema columns *Id* and *Name* and drop them on the Agency node.

The [Selection] dialog box is displayed.

Select the **Create as sub-element of target node** option so that the two columns are linked to the two XML sub-elements of the Agency node.

5. Right-click the root node **Agency** and then select **Add Sub-element**.

In the dialog box that pops up, enter a name for the new sub-element, *Offices* in this example. Repeat the same procedure to create a new sub-element **Office** for the **Offices** node which corresponds to the multi-occurrence element **Offices** of the business entity **Agency**.

6. In the Linker source area, select the schema column whose corresponding data entry you want to remove, *Remove_Office* in this example, and drop it on the new **Office** node.

The [Selection] dialog box is displayed.

Select the **Create as sub-element of target node** option so that the column is linked to the XML sub-element of the **Offices** node.

7. Click **Ok** to proceed to the next step.

8. Right-click the element in the Link Target area you want to set as a loop element and select **Set As Loop Element** from the contextual menu.

   In this example, **Id** is the iterating object.

9. Click **OK** to validate your changes and close the dialog box.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.

2. Execute the Job by pressing **F6** or clicking **Run** on the Run tab.

   The agency office located in **Paris** with the agency Id **PA05** is removed from the **Agency** business entity in the **DStar** data container on the MDM server.
### Master Data Container Browser DStar 0.1

<table>
<thead>
<tr>
<th>Date</th>
<th>Entity</th>
<th>Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>20160120 13:53:29</td>
<td>Agency</td>
<td>UT05</td>
</tr>
<tr>
<td>20160120 13:53:58</td>
<td>Agency</td>
<td>NV01</td>
</tr>
<tr>
<td>20160120 15:10:48</td>
<td>Agency</td>
<td>PA05</td>
</tr>
</tbody>
</table>

#### XML Editor/Viewer

- **Tree**
  - Agency
    - Id: PA05
    - Name: Talend
    - Offices
      - Office: Suresnes
      - Office: Beijing

- **Options**
  - Fire a change event (update report) and triggers
  - Enable verification by before processes

- **Data Model**
  - DStar

- **Buttons**
  - Save
  - Cancel
**tMDMReceive**

Decodes a context parameter holding MDM XML data and transforms it into a flat schema.
tMDMReceive receives an MDM record in XML from MDM triggers or MDM processes.

**tMDMReceive Standard properties**

These properties are used to configure tMDMReceive running in the Standard Job framework.
The Standard tMDMReceive component belongs to the Talend MDM family.
The component in this framework is generally available.

**Basic Settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where properties are stored. The fields that follow are completed automatically using the fetched data.</td>
</tr>
</tbody>
</table>

**Schema and Edit Schema**

A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th></th>
<th>Built-in: The schema will be created and stored for this component only. Related Topic: see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Repository: The schema already exists and is stored in the repository. You can reuse it in various projects and jobs. Related Topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

**XML Record**

Enter the context parameter allowing to retrieve the last changes made to the MDM server. For more information
about creating and using a context parameter, see *Talend Studio User Guide*.

**XPath Prefix**

If required, select from the list the looping xpath expression which is a concatenation of the prefix + looping xpath.

- **/item**: select this xpath prefix when the component receives the record from a process because processes encapsulate the record within an item element only.
- **/exchange/item**: select this xpath prefix when the component receives the record from a trigger because triggers encapsulate the record within an item element which is within an exchange element.

**Loop XPath query**

Set the XML structure node on which the loop is based.

**Mapping**

- **Column**: reflects the schema as defined in the *Edit schema* editor.
- **XPath query**: Type in the name of the fields to extract from the input XML structure.
- **Get Nodes**: Select this check box to retrieve the XML node together with the data.

**Limit**

Maximum number of rows to be processed. If Limit = 0, no row is read or processed.

**Die on error**

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a *Row > Reject* link.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the processing metadata at the Job level as well as at each component level.

### Global Variables

**Global Variables**

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.
- **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*. 
Usage

| Usage rule | Use this component as a start component. It needs an output flow. |

Scenario: Extracting information from an MDM record in XML

This scenario applies only to a Talend solution with MDM.

The following scenario describes a simple Job which will extract the information of interest from an MDM record in XML and display it on the console.

Scenario prerequisites

A data container Product and a data model Product are created and deployed to MDM server. The Product and Store data entities are defined and some data records already exist in them.

The entities Product and Store are linked by a foreign key which is the Name of the Store.

This example is designed to obtain the store information for a new product. Therefore, when you create a new Product record, make sure that the Store information is also added for the new Product record.

The entities and their attributes are shown below.

For more information about MDM working principles, see the MDM part in Talend Studio User Guide.
Dropping and linking the components

Procedure
1. Drop the following components from the Palette onto the design workspace: tMDMReceive, and tLogRow.
2. Connect tMDMReceive to tLogRow using a Row > Main link.
3. Rename the components to better identify their functions.

Configuring the components

Defining a context variable

Procedure
1. From the Contexts tab, click the [+] button to add one variable and name it exchangeMessage.
2. Fill in the variable value in the Value field.

Note that the XML record must conform to a particular schema. For more information about the schema, see the description of processes and schemas used in MDM processes to call Jobs in Talend Studio User Guide.
One sample of XML record from the Update Report is as follows:

```xml
<exchange xmlns:mdm="java:com.amalto.core.plugin.base.xslt.MdmExtension">
  <report>
    <Update>
      <UserName>administrator</UserName>
      <Source>genericUI</Source>
      <TimeInMillis>1381486872930</TimeInMillis>
      <OperationType>ACTION</OperationType>
      <RevisionID>null</RevisionID>
      <DataCluster>Product</DataCluster>
      <DataModel>Product</DataModel>
      <Concept>Product</Concept>
      <Key>2</Key>
    </Update>
  </report>
  <item>
    <Product>
      <Id>001</Id>
      <Name>Computer</Name>
      <Description>Laptop series</Description>
      <Availability>true</Availability>
      <Price>400</Price>
      <OnlineStore>TalendShop@http://www.cafepress.com/Talend.2</OnlineStore>
      <Stores>
        <Store>[Dell]</Store>
        <Store>[Lenovo]</Store>
      </Stores>
    </Product>
  </item>
</exchange>
```

In this example, the XML record is trimmed like this:

```xml
<exchange><report/><item><Product><Id>001</Id><Name>Computer</Name><Description>Laptop series</Description><Availability>true</Availability><Price>400</Price><OnlineStore>TalendShop@http://www.cafepress.com/Talend.2</OnlineStore><Stores><Store>[Dell]</Store><Store>[Lenovo]</Store></Stores></Product></item></exchange>
```

3. Press **Ctrl+S** to save your changes.

### Configuring tMDMReceive and tLogRow

#### Procedure

1. Double-click the **tMDMReceive** component to open its **Basic settings** view in the **Component** tab.
2. Click the [...] button next to **Edit schema** to define the desired data structure. In this example, three columns are added: **Product_ID**, **Product_Name**, and **Store_Name**.

3. In the **XML Record** field, fill in the context variable `context.exchangeMessage`.

4. From the **XPath Prefix** list, select `'/exchange/item'`.

5. In the **Loop XPath query** field, type in the name of the XML tree root tag. In this example, type in `'/Product/Stores/Store'`.

6. The **Column** column in the **Mapping** table is populated with the columns defined in the schema. In the **XPath query** column, enter the XPath query accordingly. In this example, the information of product ID, product name and store name will be extracted.

7. Double-click the **tLogRow** component to open its **Basic settings** view in the **Component** tab.

8. Select **Table (print values in cells of a table)** in the **Mode** area.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.

2. Execute the Job by pressing **F6** or clicking **Run** on the **Run** tab.

   The product information of interest extracted from the XML record is displayed on the console.

   ```
   [statistics] connecting to socket on port 3383
   [statistics] connected
   ___________________________
<p>| Product_Info |</p>
<table>
<thead>
<tr>
<th>Product_ID</th>
<th>Product_Name</th>
<th>Store_Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Computer</td>
<td>[Dell]</td>
</tr>
<tr>
<td>001</td>
<td>Computer</td>
<td>[Lenovo]</td>
</tr>
</tbody>
</table>
   ___________________________
   [statistics] disconnected
   Job Product_info ended at 17:48 11/10/2013. [exit code=0]
   ```
**tMDMRollback**

Rolls back any changes made in the database rather than definitively committing them, for example to prevent partial commits if an error occurs.

`tMDMRollback` returns a database to its original state before a Job was run, instead of committing any changes.

**tMDMRollback Standard properties**

These properties are used to configure `tMDMRollback` running in the Standard Job framework.

The Standard `tMDMRollback` component belongs to the Talend MDM family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Component List</th>
<th>Select the <code>tMDMConnection</code> component for which you want the rollback action to be performed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close Connection</td>
<td>Select this check box to close the session for this connection to the MDM Server after rolling back the changes. Note that even if you do not select this check box, the connection can still not be used in a subsequent subjob unless the Autocommit mode has been enabled.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | `ERROR_MESSAGE`: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the `Die on error` check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press `Ctrl + Space` to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

For further information about variables, see *Talend Studio User Guide*. |
Usage

| Usage rule | This component is to be used along with the tMDMConnection, tMDMCommit, tMDMSP, tMDMViewSearch, tMDMInput, tMDMDelete, tMDMRouteRecord, tMDMOutput, and tMDMClose components. |

Related scenario

For a related use case, see Scenario: Deleting master data from an MDM Hub on page 2138.
tMDMRouteRecord

Helps Event Manager to identify the changes you have made on your data so that correlative actions can be triggered.

tMDMRouteRecord submits the primary key of a record stored in your MDM Hub to Event Manager in order for Event Manager to trigger the due process(es) against some specific conditions that you can define in the process or trigger pages of the MDM Studio.

For more information on Event Manager, see Talend Studio User Guide.

**tMDMRouteRecord Standard properties**

These properties are used to configure tMDMRouteRecord running in the Standard Job framework.
The Standard tMDMRouteRecord component belongs to the Talend MDM family.
The component in this framework is generally available.

**Basic Settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection</td>
<td>Select this check box if you want to use a configured tMDMConnection component.</td>
</tr>
<tr>
<td>MDM version</td>
<td>By default, Server 6.0 is selected. Although it is recommended to migrate existing Jobs for this new version, the Server 5.6 option is available to ease the process of the migration of your Jobs so as to keep them working without modification with a 6.0 server. To do so, an option on the server must be enabled to accept and translate requests from such Jobs.</td>
</tr>
<tr>
<td>URL</td>
<td>Type in the URL of the MDM server.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Type in the user authentication data for the MDM server. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Data Container</td>
<td>Type in the name of the data container that holds the record you want Event Manager to read.</td>
</tr>
<tr>
<td>Type</td>
<td>Select Master or Staging to specify the database on which the action should be performed.</td>
</tr>
<tr>
<td>Entity Name</td>
<td>Type in the name of the business entity that holds the record you want Event Manager to read.</td>
</tr>
<tr>
<td>IDs</td>
<td>Specify the primary key(s) of the record(s) you want Event Manager to read.</td>
</tr>
</tbody>
</table>
Advanced settings

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

Usage

| Usage rule | Use this component as a start component. It needs an output flow. You can increase the timeout values for a Job using this component to help process a large number of data records. For more information, see advanced execution settings for JVM parameters on Talend Help Center ([https://help.talend.com](https://help.talend.com)). |

Connections

| Connections | Outgoing links (from this component to another): **Row**: Iterate **Trigger**: Run if; On Component Ok; On Component Error, On Subjob Ok, On Subjob Error. Incoming links (from one component to this one): **Row**: Iterate; **Trigger**: Run if, On Component Ok, On Component Error, On Subjob Ok, On Subjob Error For further information regarding connections, see *Talend Studio User Guide*. |

Scenario: Routing an update report record to Event Manager

This scenario applies only to a Talend solution with MDM.
In this scenario, the tMDMRouteRecord component is used to submit the primary key of an update report record to Event Manager so that a trigger is launched to call a Job which will print out the notification that a data record had been updated.

Prerequisites

- The MDM server is up and running.
- The MDM demo project has already been imported.
- The data container Product has already been populated with some data records.

For further information about how to create a data container and a data model, see your Talend Studio User Guide.

For further information about how to create a record and access its viewable attributes, see the Talend MDM Web UI User Guide.

Creating a Job to notify users that a data record has been updated

In this scenario, a Job called message is created with only one tJava component to inform the users of the update of an existing data record.

Procedure

1. Drop a tJava component onto the design workspace.
2. Double-click the component to display its Basic settings.
3. In the Code field, enter `System.out.println("------- A Product is Updated.--" + new java.util.Date());`.
4. Save your Job and deploy it to the MDM Server.

Results

For further information about the tJava component, see tJava on page 1834.

For further information about how to deploy a Job onto the MDM Server, see the Talend Studio User Guide.

Generating a Job Caller Trigger and deploying it to the MDM server

In this scenario, once the Job message is created, a trigger calling this Job can be generated.

Procedure

1. In the MDM Repository tree view, under the Job Designs node, right click the message Job.
2. In the contextual menu, select Generate Talend Job Caller Trigger, accept the default options and click Generate.
   
   A trigger used to call the message Job is generated and stored under Event Management > Trigger. Its name is CallJob_Message, and its Service JNDI Name is callJob.
3. In the Trigger xPath Expressions area, customize the conditions of the trigger according to your needs.
   
   In this example, add the following conditions:
4. In the **Conditions** area, combine the conditions using the predicates.

In this example, enter the following:

```
C1 AND C2 AND C3
```

In this way, this Trigger will be executed on a record in the Update entity only when a Product data record has been updated.

5. Deploy the customized Trigger to the MDM server.

**Updating a data record in a specific data container**

**Procedure**

1. Log onto your Talend MDM Web UI.

   For further details about how to log onto the Talend MDM Web UI, see *Talend MDM Web UI User Guide*.

2. In the upper right corner of the web page, make sure the **Actions** panel is displayed.

3. In the **Domain Configuration** area, select the required data container and data model.

   In this example, select Product for both the data container and the data model.

4. Click **Save** to save your changes.

5. Go to the **Master Data Browser** page, and select the Product entity from the list.

6. Double-click one of the product records to display its viewable attributes in a new view dedicated to this product.

   In this example, open the product record **Talend Mug** with the Unique Id of **231035938**.

7. Update one of the attribute values in the product record.

   In this example, update the record by selecting the **Availability** check box to make it available.

8. Click **Save** to validate this update.

**Accessing the Update Reports**

**Procedure**

1. In the **MDM Repository** tree view, double click Data Container > System > UpdateReport to open the UpdateReport view.

   An Update Report is a complete track of all create, update or delete actions on any master data.

   If the UpdateReport data container is not available, you may first have to import it from your MDM Server. For details of how to import system objects from the MDM Server to your local repository, see the Talend Studio User Guide.
2. Next to the Entity field of this view, click the button to search all the action records in the UpdateReport.

Note that the Update entity does not necessarily mean that the corresponding action recorded is update, since it is just the entity name defined by the data model of UpdateReport and may record different actions including create, delete, and update.

3. Locate the record that corresponds to your action of updating the data record as described in Updating a data record in a specific data container on page 2186.

The update report record will be routed to Event Manager, whose primary key is genericUI.1499236505686.

Setting up the Job to route the update report record to Event Manager

Procedure

1. In the Integration perspective, right-click Job Designs in the Repository tree view and select Create Standard Job in the contextual menu.
2. Enter RouteRecord in the Name field in the wizard that opens, and then click Finish.
3. Drop the tMDMRouteRecord component from the Palette onto the design workspace.
4. Double click this component to open its Component view.
5. In the URL field, enter the URL for accessing your MDM server.
   In this example, enter http://localhost:8180/talendmdm/services/soap.
6. In the Username and the Password fields, enter the credentials for accessing the MDM server.
7. In the Data Container field, enter the data container name that stores the record you want to route.
   It is UpdateReport in this example.
8. In the Entity Name field, enter the entity name that the record you want to route belongs to.
   It is Update in this example.
9. In the IDS area, click the plus button under the table to add a new line.
10. In the newly added line, fill in the primary key of the update report record to be routed to Event Manager, which was read earlier from the UpdateReport data container.
    In this example, the primary key is genericUI.1499236505686.
**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to run this Job.

   In this example, since the Product data record with the Unique Id of 231035938 is updated, the component **tMDMRouteRecord** submits the primary key of the corresponding update report record to the **Event Manager**. When the **Event Manager** checks this record and confirms that this record meets the conditions in the trigger **CallJob_Message**'s configuration view, the **Event Manager** calls the trigger to invoke the **message Job**.

3. Verify the execution results of this Job by consulting the MDM server log to see that the notification `------- A Product is Updated.--` is printed out.

   You can check the server log in one of the following ways:

   - In the **Server Explorer** panel, right-click the connection to the server where MDM server is installed, and then click **View Server Log**. A Console opens displaying the most recent contents of the **mdm.log** file.

   - You can also view the contents of the **mdm.log** file in a browser. For more information, see the Talend Studio User Guide.

   - If you have direct access to the MDM server, you can also consult the file directly in the `<$INSTALLDIR>/log` directory, with **INSTALLDIR** indicating where the MDM server is installed.
tMDMSP

Offers a convenient way to centralize multiple or complex queries in an MDM Hub and calls the stored procedure easily.

tMDMSP Standard properties

These properties are used to configure tMDMSP running in the Standard Job framework.
The Standard tMDMSP component belongs to the Talend MDM family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>In SP principle, the schema is an input parameter. A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema of this component is read-only.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box if you want to use a configured tMDMConnection component.</td>
</tr>
<tr>
<td><strong>MDM version</strong></td>
<td>By default, <strong>Server 6.0</strong> is selected. Although it is recommended to migrate existing Jobs for this new version, the <strong>Server 5.6</strong> option is available to ease the process of the migration of your Jobs so as to keep them working without modification with a 6.0 server. To do so, an option on the server must be enabled to accept and translate requests from such Jobs.</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td>Type in the URL of the MDM server.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Type in the user authentication data for the MDM server. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Data Container</strong></td>
<td>Type in the name of the data container that stores the procedure you want to call.</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Select Master or Staging to specify the database on which the action should be performed.</td>
</tr>
<tr>
<td><strong>Procedure Name</strong></td>
<td>Type in the exact name of the Stored Procedure.</td>
</tr>
<tr>
<td><strong>Parameters (in order)</strong></td>
<td>Click the Plus button and select the various Input Columns that will be required by the procedures.</td>
</tr>
</tbody>
</table>

*Note:*
The SP schema can hold more columns than there are parameters used in the procedure.
**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

**Usage**

| Usage rule | This component is used as intermediary component. It can be used as start component but only no input parameters are thus needed for the procedure to be called. An output link is required. You can increase the timeout values for a Job using this component to help process a large number of data records. For more information, see advanced execution settings for JVM parameters on Talend Help Center ([https://help.talend.com](https://help.talend.com)). |

**Connections**

| Connections | Outgoing links (from this component to another): **Row**: Main **Trigger**: Run if; On Component Ok; On Component Error, On Subjob Ok, On Subjob Error. Incoming links (from one component to this one): **Row**: Main, Iterate; **Trigger**: Run if, On Component Ok, On Component Error, On Subjob Ok, On Subjob Error For further information regarding connections, see Talend Studio User Guide. |

**Executing a stored procedure using tMDMSP**

This scenario applies only to a Talend solution with MDM.
In this scenario, the Job first generates parameters and sends them to tMDMSP, which executes a predefined stored procedure, and then extracts data from the returned execution result and presents the extracted data in the console.

![Diagram of the Job flow](image)

**Prerequisites:**
- Make sure the MDM server is up and running.
- You have imported the MDM demo project and loaded the sample data into the data container Product by running the Job MDM_LoadAll.
- You have created a store procedure called ProductSelection and deployed this stored procedure to the MDM server.

In this example, the stored procedure ProductSelection is designed to query two fields Name and Price of Product data records within a price range:

![Image of the stored procedure](image)

For more information on working with stored procedures, see [Talend Studio User Guide](https://docs.talend.com).

**Creating a Job to execute the stored procedure**

**Procedure**
1. Drag and drop the following components onto the design workspace: tFixedFlowInput, tMDMSP, tExtractXMLField, and tLogRow.
2. Link the components using the Row > Main connections.

**Configuring tFixedFlowInput to generate a price range**

**Procedure**
1. Double-click tFixedFlowInput to open its Basic settings view.
2. Click the [...] button next to Edit schema to open the schema editor.
3. Click the + button to add two parameters min and max that are used to define the price range.

4. Click OK to close the schema editor.
5. In the Mode area, keep the default option Use Single Table, and enter 10 between quotes for the min parameter and 15 between quotes for the max parameter.

**Configuring tMDMSP to execute the stored procedure**

**Procedure**

1. Double-click tMDMSP to open its Basic settings view.
2. Enter the user name and password for accessing the MDM server.

3. In the **Data Container** field, enter *Product* between quotes.

4. In the **Procedure Name** field, enter *ProductSelection* between quotes.

5. In the **Parameters** table, click the [+] button to add two rows, and select *min* and *max* respectively.

**Configuring tExtractXMLField to extract data from the returned execution result of the stored procedure**

**Procedure**

1. Double-click **tExtractXMLField** to open its **Basic settings** view.

2. Click the [...] button next to **Edit schema** to open the schema editor.
3. Add two columns to define the structure of the output data: **name** and **price**.

![Schema editor](image)

4. Click **OK** to close the schema editor, and then click **Yes** in the [Propagate] dialog box.

5. In the **Loop XPath query** field, enter **result** between quotes on which the loop is based.

   Each result of the stored procedure has the following syntax: `<result><col0>val1</col1>...<colN>valN</colN></result>` (where **colN** is the **n**th column in the **SELECT** clause).

   **Warning:**
   
   If a null value exists in a data record to be returned by the execution result of the stored procedure, some inconsistencies will occur between the column number and the value for this column in the returned data record.

6. In **XPath query** column of the **Mapping** table, enter the exact node name on which the loop is applied: **col0** and **col1**, respectively, between quotes.

### Configuring the data display mode and executing the Job

**Procedure**

1. Double-click the **tLogRow** component to display its **Basic settings** view.

2. In the **Mode** area, select **Table (print values in cells of a table)** for better readability of the result.

   ![Basic settings](image)

3. Save the Job and press **F6** to run it.
According to the stored procedure and the price range, the *Product* data records within the price range of 10 to 15 are displayed with the values of the *Name* and *Price* fields as expected.

<table>
<thead>
<tr>
<th>tLogRow_1</th>
<th>name</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Talend Trucker Hat</td>
<td>10.93</td>
</tr>
<tr>
<td></td>
<td>Talend Mug</td>
<td>10.93</td>
</tr>
<tr>
<td></td>
<td>Talend Large Mug</td>
<td>11.99</td>
</tr>
<tr>
<td></td>
<td>Talend Stein</td>
<td>13.99</td>
</tr>
<tr>
<td></td>
<td>Talend Cap</td>
<td>14.99</td>
</tr>
</tbody>
</table>
tMDMTriggerInput

Reads the XML message (Document type) sent by MDM and passes the information to the component that follows.

Every time when you save a change in your MDM, the corresponding record is generated in XML format. At runtime, this component reads this record and sends the relative information to the following component. With tMDMTriggerInput, you do not need to configure your Job any more in order to communicate the data changes from MDM to your Job.

Note:
This component works alongside the new trigger service and process plug-in in MDM version 5.0 and higher. The MDM Jobs, triggers and processes developed in previous MDM versions remain supported. However, we recommend using this component when designing new MDM Jobs.

tMDMTriggerInput Standard properties

These properties are used to configure tMDMTriggerInput running in the Standard Job framework.

The Standard tMDMTriggerInput component belongs to the Talend MDM family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in:</strong> No property data stored centrally. tMDMTriggerInput is expected to use this option in order to apply the default read-only schema. MDM_message is the only column of this schema.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> Select the repository file where properties are stored. The fields that follow are completed automatically using the fetched data. As tMDMTriggerInput provides a fixed read-only schema, you are expected to use the Built-in option.</td>
</tr>
</tbody>
</table>

Schema and Edit Schema

A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to Built-in for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and
decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in**: The schema will be created and stored for this component only. Related Topic: see Talend Studio User Guide.

This is the default option for tMDMTriggerInput. With this option, the read-only schema is used to deal with the XML-format MDM message.

**Repository**: The schema already exists and is stored in the repository. You can reuse it in various projects and jobs. Related Topic: see Talend Studio User Guide.

As tMDMTriggerInput provides a fixed read-only schema, you are expected to use the **Built-in** option.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the processing metadata at the Job level as well as at each component level.

### Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

**Usage rule**

Use this component as a start component. It needs an output flow.

To receive the message from MDM, you need to deploy the Job using this component on your MDM server and generate the corresponding trigger and process in MDM to invoke this Job.

For further information about how to deploy a Job onto MDM server and how to generate a trigger or a process, see Talend Studio User Guide.
Scenario: Exchanging the event information about an MDM record

This scenario applies only to a Talend solution with MDM.

In this scenario, a four-component Job is used to exchange the event information about a product record. Using an MDM connection established from the Repository, this Job is called by an eligible trigger once you update an existing product record.

To replicate this scenario, accomplish the following tasks sequentially:

1. Create an MDM connection of the Receive type in the Repository of the Studio. This connection is to the MDM hub holding the record you want to update.

2. Create the Job to receive and send the MDM update message.

3. Generate the trigger to invoke this Job created.

4. Update a product record.

To create the MDM records, data model and data container used in this scenario, you can execute the Jobs in the MDM demo project in Talend Studio and then update the MDM server to deploy the objects thus created for them to be taken into account at runtime. You will use this server throughout this scenario.

You can search for further information about MDM event management, how to import a demo project and how to deploy objects to the MDM server on Talend Help Center (https://help.talend.com).

Creating an MDM connection to retrieve entities

Establishing the connection

Before you begin

Make sure the MDM server with which you need to communicate the update message is up and running.

Procedure

1. In the Integration perspective of Talend Studio, expand the Metadata node in the Repository.

2. Right-click the Talend MDM item and select Create MDM connection.
3. Enter the **Name** you want to use for this connection and if required, added the **Purpose** and the **Description** in the corresponding fields.

   For example, name this connection **Receive**.

4. In the **Next** step, enter the credentials for connecting to the MDM server which holds the record to be updated.

   Once you click the **Check** button and the connection is shown successful, the **Next** button becomes clickable.
5. In the Next step, select the Data model and the Data Container used by the record to be updated.
   In this scenario, the model and the container are both Product.
6. Click Finish to validate the creation.
   The connection created is displayed under the Metadata node in the Repository.

**Retrieving entities**

**Procedure**

1. Right-click the connection created and from the contextual menu, select Retrieve Entity.
2. Select Receive MDM in the dialog box and click Next to continue.
3. Select the entity to be retrieved, and click Next.
   In this scenario, select Product, and the Name field is populated automatically.

4. Drop the elements you need to retrieve from the Source Schema area to the Target Schema area.
   In this scenario, the Features element is the field to be looped and the Id, Name and Description elements are the fields to be extracted.
5. Keep the default schema description and click **Finish** to finalize retrieving the schema of the Product entity.

You can see more information about the data container and the data model used by Talend MDM on Talend Help Center (https://help.talend.com).

**Creating the Job communicating the MDM message**

**Linking the components**

**Procedure**

1. Create a Job named **update_product**.
2. Drop **tMDMTriggerInput**, **tXMLMap**, **tMDMTriggerOutput** and **tLogRow** from **Palette** onto the workspace.
3. Right-click **tMDMTriggerInput** and from the contextual menu, select the **Row > Main** link to connect it to **tXMLMap**.

4. Do the same to connect **tXMLMap** to **tMDMTriggerOutput**, and name the connection output.

5. Double-click **tMDMTriggerOutput** to open its **Component** view.

6. Click the [...] button next to **Edit schema** to open the editor.

7. Select the single pre-defined column of **tMDMTriggerOutput**, and then click ![button] to reproduce this column on the input side (left).

8. Connect **tMDMTriggerOutput** to **tLogRow** using the **Row > Main** connection.

![Diagram of data flow]

**Configuring the transformation of the MDM message**

**Procedure**

1. Double-click **tXMLMap** to open its editor.

2. In the table representing the input flow (upper left of the editor), right-click the column name **MDM_Message** on the top of the XML tree and select **Import From Repository**.

   The **Metadata** wizard is displayed.

3. Select the entity schema **ProductReceive** retrieved earlier using the **Receive MDM** model, then click **OK**.

4. Click **OK** in the dialog box that prompts you to add the schema of the Update Report to the input XML tree.

   This builds a complete input document for an MDM event. In the input XML tree, the **Features** element is set as the loop element automatically.
5. In the table representing the output flow (upper right of the editor), develop the output XML tree as presented in the figure below.

This tree is constructed depending on the required static model of the MDM output report.

The XML construct required to return the validation-success message is \(<report><message type="info">message</message></report>\), and the XML construct required to return the validation-failure message is \(<report><message type="error">message</message></report>\).
6. Map the **OperationType** element on the input side to the **message** element on the output side.

   This will output the information about the type of the event occurring on the MDM record.

7. In the **Expression** column, enter "info" in the row corresponding to @type.

8. Click the pincer icon to display the output settings panel, and then set the **All in one** option to true.

9. Click **OK** to close the editor and validate these changes.

10. Double click **tLogRow** to open its **Component** view, then, click **Sync columns**.

### Generating the trigger to invoke the Job created

### Deploying the Job to be deployed onto the MDM server

**Procedure**

1. Switch to the **MDM** perspective, and expand the **Job Designs** node in the **MDM Repository** tree view.

2. Right-click the Job **update_product** created previously in this scenario, and select **Deploy To...** from the contextual menu.

   The deployment wizard is displayed.

3. From the server list, select the MDM server you are using, and then click **OK**.

4. In the **Deploy to Talend MDM** window that pops up, keep the default settings: **Export type** is **Hosted (zip)** and **Context scripts** is **Default**.

   You can see more information about these settings on Talend Help Center ([https://help.talend.com](https://help.talend.com)).

5. Click **Finish** to validate these settings and start the deployment.

   When the deployment is done, a message box pops up to indicate that the deployment is successful.

6. Click **OK** to close this message box.

   A window pops up to indicate that the Job **update_product** is deployed successfully.

7. Click **OK** to finalize the deployment.
Generating the trigger used to call the Job

Procedure

1. Right-click the Job `update_product` and select Generate Talend Job Caller Trigger from the contextual menu.

2. In the pop-up window, keep the default settings for this scenario: Integrated and Embedded.

3. Click Generate to start the generation.
   
   Once done, a trigger named `CallJob_update_product` is displayed under the Trigger node in MDM Repository.

4. Right-click this trigger, and then select Deploy To... from the contextual menu to deploy it onto the MDM server.

5. In the pop-up wizard, select the server you are using, and click OK.

6. Click OK to finalize the deployment.

Updating a product record

Procedure

1. Log in to the web user interface.

2. In the Actions panel on the right side, select Product for both the Data Container and the Data Model.

3. In the Master Data Browser page, launch the search in the Product entity to list all the available product records.

4. Select the product record you need to update from the list, for example, Talend Trucker Hat. The details of this record are displayed in the Product tab view.

5. Update one of its attributes.
   
   In this example, update the price to 11.00 and then click Save.

   The message about the operation type of this event has been sent to the MDM server and this message is displayed on the window of this MDM server.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root><report><message type="info">UPDATE</message></report></root>
```
tMDMTriggerOutput

Receives an XML flow (Document type) from the preceding component in the Job.

tMDMTriggerOutput receives an XML flow to set an MDM message so that MDM retrieves this message at runtime. With this component, you do not need to configure your Job any more in order to communicate the data changes from MDM to your Job.

Note:
This component works alongside the new trigger service and process plug-in in MDM version 5.0 and higher. The MDM Jobs, triggers and processes developed in previous MDM versions remain supported. However, we recommend using this component when designing new MDM Jobs.

tMDMTriggerOutput Standard properties

These properties are used to configure tMDMTriggerOutput running in the Standard Job framework.
The Standard tMDMTriggerOutput component belongs to the Talend MDM family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in: No property data stored centrally. tMDMTriggerOutput is expected to use this option in order to apply the default read-only schema. MDM_message is the only column of this schema.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file where properties are stored. The fields that follow are completed automatically using the fetched data. As tMDMTriggerOutput provides a fixed read-only schema, you are expected to use the Built-in option.</td>
<td></td>
</tr>
</tbody>
</table>

Schema and Edit Schema

A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- View schema: choose this option to view the schema only.
- Change to built-in property: choose this option to change the schema to Built-in for local changes.
- Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate
the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th><strong>Built-in:</strong> The schema will be created and stored for this component only. Related Topic: see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the default option for tMDMTriggerOutput. With this option, the read-only schema is used to deal with the XML-format MDM message.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Repository:</strong> The schema already exists and is stored in the repository. You can reuse it in various projects and jobs. Related Topic: see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td>As tMDMTriggerOutput provides a fixed read-only schema, you are expected to use the Built-in option.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

**Usage**

| Usage rule | Use this component as an end component. It needs an input flow. To send message to MDM, you need to deploy the Job using this component on your MDM server and generate the corresponding trigger and process to invoke this Job in MDM. For further information about how to deploy a Job onto MDM server and how to generate a trigger or a process, see Talend Studio User Guide. |
Related scenario

For a related scenario, see Scenario: Exchanging the event information about an MDM record on page 2198
tMDMViewSearch

Retrieves the MDM records from an MDM hub by applying filtering criteria you have created in a specific view and puts out results in XML structure.

For more information on a view on which you can define filtering criteria, see *Talend Studio User Guide*.

### tMDMViewSearch Standard properties

These properties are used to configure tMDMViewSearch running in the Standard Job framework.

The Standard tMDMViewSearch component belongs to the Talend MDM family.

The component in this framework is generally available.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either **Built-in** or remote in the **Repository**. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  • **View schema**: choose this option to view the schema only.  
  • **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.  
  
  Click **Sync columns** to collect the schema from the previous component. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: You create the schema and store it locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>XML Field</strong></td>
<td>Select the name of the column in which you want to write the XML data.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box if you want to use a configured tMDMConnection component.</td>
</tr>
<tr>
<td><strong>MDM version</strong></td>
<td>By default, <strong>Server 6.0</strong> is selected. Although it is recommended to migrate existing Jobs for this new version, the <strong>Server 5.6</strong> option is available to ease the process of the migration of your Jobs so as to keep them working without modification with a 6.0 server. To do so, an option on the server must be enabled to accept and translate requests from such Jobs.</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td>Type in the URL of the MDM server.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Type in the user authentication data for the MDM server. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Data Container</strong></td>
<td>Type in the name of the data container that holds the master data you want to read.</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Select <strong>Master</strong> or <strong>Staging</strong> to specify the database on which the action should be performed.</td>
</tr>
<tr>
<td><strong>View Name</strong></td>
<td>Type in the name of the view whose filters will be applied to process the records.</td>
</tr>
<tr>
<td><strong>Operations</strong></td>
<td>Complete this table to create the WHERE clause. The parameters to be set are:</td>
</tr>
<tr>
<td></td>
<td>- <strong>XPath</strong>: define the path expression to select the XML node at which point the filtering is operated.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Functions</strong>: select an operator from the drop-down list, like <strong>Contains</strong>, <strong>Starts with</strong>, <strong>Equals</strong>, etc.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Value</strong>: enter the value you want to retrieve. The value can be a string (for example, &quot;Apple&quot;), an Entity element (for example, &quot;Product/Name&quot;) or a user-based variable (for example, &quot;$[user_context.language&quot; or &quot;$[user_context.properties['location']]&quot;).</td>
</tr>
<tr>
<td></td>
<td>- <strong>Predicate</strong>: select the predicate to combine the filtering conditions in different manners. The predicate may be <strong>none</strong>, <strong>or</strong>, <strong>and</strong>, <strong>exactly</strong>, etc.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning</strong>: The parameters are case sensitive. For more information about the operators and the predicates, see tMDMInput Standard properties on page 2145.</td>
</tr>
<tr>
<td><strong>Order (One Row)</strong></td>
<td>Complete this table to decide the presentation order of the retrieved records. The parameters to be set are:</td>
</tr>
<tr>
<td></td>
<td>- <strong>XPath</strong>: define the path expression to select the XML node at which point the sorting operation is performed.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Order</strong>: select the presentation order that may be <strong>asc</strong> (ascending) or <strong>desc</strong> (descending).</td>
</tr>
<tr>
<td></td>
<td><strong>Warning</strong>: The parameters are case sensitive.</td>
</tr>
<tr>
<td>Spell Threshold</td>
<td>Set it to -1 to deactivate this threshold. This threshold is used to decide the spell checking level.</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Skip Rows</td>
<td>Type in the count of rows to be ignored to specify from which row the process should begin. For example, if you type 8 in the field, the process will begin from the 9th row.</td>
</tr>
<tr>
<td>Max Rows</td>
<td>Type in the maximum number of rows to be processed. If Limit = 0, no row is read or processed. By default, the limit is -1, meaning that no limit is set.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Number of lines in each processed batch. By default, the batch size is set to -1, meaning that all the lines are processed in one batch.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

**NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

**Usage rule**

Use this component to retrieve specific records.

You can increase the timeout values for a Job using this component to help process a large number of data records. For more information, see advanced execution settings for JVM parameters on Talend Help Center (https://help.talend.com).
Scenario: Retrieving records from an MDM hub via an existing view

This scenario applies only to a Talend solution with MDM.

This scenario describes a two-component Job that retrieves a data record in XML structure.

In this example, you will select the T-shirt information from the Product entity via the Browse_items_Product view created from Talend Studio. Each record in the entity contains the details defined as filtering criteria: Id, Name, Description and Price.

- From the Palette, drop tMDMViewSearch and tLogRow onto the design workspace.
- Connect the components using a Row Main link.
- Double-click tMDMViewSearch to view its Basic settings, in the Component tab and set the component properties.
In the **Schema** list, select **Built-In** and click the three-dot button next to **Edit schema** to open a dialog box in which you can define the structure of the XML data you want to write in.

Click the plus button and add one column of the type **String**. Name the column as **Tshirt**.

Click **OK** to validate your creation and proceed to the next step.

In the **XML Field** field, select **Tshirt** as the column you will write the retrieved data in.

Use your MDM server address in the **URL** field and type in the corresponding connection data in the **Username** and the **Password** fields. In this example, use the default url, then enter *admin* as username as well as password.

In the **Data Container** field, type in the container name: **Product**.

In the **View Name** field, type in the view name: **Browse_item_Product**.
• Below the Operations table, click the plus button to add one row in this table.

• In the Operations table, define the XPath as Product/Name, meaning that the filtering operation will be performed at the Name node, then select Contains in the Function column and type in Tshirt in the Value column.

• Below the Order (One Row) table, click the plus button to add one row in this table.

• In the Order (One Row) table, define the XPath as Product/Id and select the asc order for the Order column.

• In the design workspace, click tLogRow to open its Basic settings view and set the properties.

• Next to the three-dot button used for editing schema, click Sync columns to acquire the schema from the preceding component.

• Press F6 to execute the Job.

```xml
<result>
  <Id>1</Id>
  <Name>Tshirt-V</Name>
  <Description>green</Description>
  <Price>16.00</Price>
</result>
<result>
  <Id>2</Id>
  <Name>Tshirt-G</Name>
  <Description>Green</Description>
  <Price>17.00</Price>
</result>
```

In the console docked in the Run view, you can read the retrieved Tshirt records in XML structure, which are sorted in the ascending order.
tMemorizeRows

Memorizes a sequence of rows that passes through and allows the following component(s) to perform operations of your choice on the memorized rows.

tMemorizeRows temporarily memorizes an array of incoming data in a row by row sequence and instantiates this array by indexing each of the memorized rows from 0. The maximum number of rows to be memorized at any given time is defined in the Basic settings view.

**tMemorizeRows Standard properties**

These properties are used to configure tMemorizeRows running in the Standard Job framework.
The Standard tMemorizeRows component belongs to the Misc family.
The component in this framework is generally available.

**Basic settings**

| Schema and Edit schema | A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository. Click Edit schema to make changes to the schema. If the current schema is of the repository type, three options are available:
|                       | • View schema: choose this option to view the schema only.
|                       | • Change to built-in property: choose this option to change the schema to Built-in for local changes.
|                       | • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.
|                       | Click Sync columns to retrieve the schema from the previous component connected in the Job. |
| Row count to memorize | Define the row count to be memorized. |
| Columns to memorize   | Select the columns to be memorized from the incoming data schema. |
## Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

## Global Variables

| Global Variables | NB_LINE_ROWS: the number of rows memorized. This is an After variable and it returns an integer.  
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

## Usage

| Usage rule | This component can be used as intermediate step in a data flow or the last step before beginning a subjob.  
Note: You can use the global variable NB_LINE_ROWS to retrieve the value of the Row count to memorize field of the tMemorizeRows component. |

| Connections | Outgoing links (from this component to another):  
Row: Main  
Trigger: Run if; On Component Ok; On Component Error.  
Incoming links (from one component to this one):  
Row: Main;  
For further information regarding connections, see Talend Studio User Guide. |

## Scenario: Retrieving the different ages and lowest age data

This scenario displays the number of occurrences of different ages and the lowest age within a group of customers. In this scenario, the customer data is entered manually.

You will see two ways to handle this data with the tMemorizeRows:

- Inside the same subjob (with the tJavaFlex)
- Outside the tMemorizeRows subjob (with the tJava)
This Job uses five components:

- **tFixedFlowInput**: it contains rows of customer data such as IDs, names and ages of the customers.
- **tSortRow**: it sorts the rows according to the age data.
- **tMemorizeRows**: it temporarily memorizes a specific number of incoming data rows at any given time and indexes the memorized data rows.
- **tJavaFlex**: it compares the age values of the data memorized by the preceding component, counts the occurrences of different ages and displays these ages in the Run view.
- **tJava**: it displays the number of occurrences of different ages and the lowest age.

To replicate this scenario, proceed as follows:

### Dropping and linking the components

**Procedure**

1. Drop a **tFixedFlowInput**, a **tSortRow**, a **tMemorizeRows**, a **tJavaFlex** and a **tJava** component by typing their names in the design workspace or dropping them from the Palette.
2. Connect the **tFixedFlowInput** component to the **tSortRow** component using a Row > Main connection.
3. Do the same to link the **tSortRow** component to the **tMemorizeRows** component and the **tMemorizeRows** component to the **tJavaFlex** component.
4. Connect the **tFixedFlowInput** component to the **tJava** component using the Trigger > OnSubjobOk connection.

### Configuring the components

**Configuring the tFixedFlowInput component**

**Procedure**

1. Double-click the **tFixedFlowInput** component to open its Basic settings view on the Component tab.
2. Click the [...] button next to **Edit schema** to open the [Schema] dialog box and define the data structure of the input data.

3. In this editor, click the [+ ] button three times to add three columns and name them **id**, **name** and **age**.

4. In the **Type** column, select **Integer** for **id** and **age**.

5. Click **OK** to close the editor, then click **Yes** to validate these changes and accept the propagation prompted by the dialog box that pops up.

6. Select **Use Inline Content (delimited file)** in the **Mode** area.

   In the **Content** field, enter the following customer data:

```
1; Judy; 27
2; Lily; 45
3; Peter; 59
4; John; 30
5; Teddy; 45
```
Configuring the **tSortRow** component

**Procedure**

1. Double-click the **tSortRow** to open its **Basic settings** view on the **Component** tab.

   ![tSortRow Component](image1)

   - **Basic settings**
   - **Criteria**
     - **Schema column**: age
     - **sort num or alpha?**: num
     - **Order asc or desc?**: desc

2. In the **Criteria** table, click the [+ ] button to add one row.

3. In the **Schema column** column, select the data column on which the sorting operation is based. In this example, select **age** as it is the ages that should be compared and counted.

4. In the **sort num or alpha?** column, select **num**, which means numerical, as **age** contains integer data.

5. In the **Order asc or desc?** column, select **desc** to display data on the console in descending order.

Configuring the **tMemorizeRows** component

**Procedure**

1. Double-click the **tMemorizeRows** component to open its **Basic settings** view on the **Component** tab.

   ![tMemorizeRows Component](image2)

   - **Row count to memorize**: 2
   - **Columns to memorize**
     - **id**
     - **name**
     - **age**

2. In the **Row count to memorize** field, type in the maximum number of rows to be memorized at any given time. As in this example you need to compare ages of two customers for each time, enter 2. This component memorizes two rows at maximum at any given moment and always indexes the new incoming row as 0 and the previous incoming row as 1.

3. In the **Memorize** column of the **Columns to memorize** table, select the check box(es) to determine the column(s) to be memorized. In this example, select the check box corresponding to **age**.
Configuring the tJavaFlex and tJava components

Procedure

1. Double-click the tJavaFlex component to open its Basic settings view on the Component tab.

2. In the Start code area, enter the Java code that will be called during the initialization phase. In this example, type `int count=0;` in order to declare a variable `count` and assign the value 0 to it.

3. In the Main code area, enter the Java code to be applied to each row in the data flow. In this scenario, type `if(!age_tMemorizeRows_1[0].equals(age_tMemorizeRows_1[1])) { count++; } System.out.println(age_tMemorizeRows_1[0]);` This code compares two ages memorized by the tMemorizeRows component each time and counts one change every time the ages are found to be different. This code then displays the age that has been indexed as 0 by the tMemorizeRows component. When the tJavaFlex component is in the same flow of the tMemorizeRows component, the variable format is ColumnName_ComponentName[index].

4. In the End code area, enter the Java code that will be called during the closing phase. In this example, type `globalMap.put("number", count);` to initialize the global variable `number` with the value of the `count` variable.
5. Double-click the **tJava** component to open its **Basic settings** view on the **Component** tab.

   ![tJava Component](image1)

   **Code**

   ```java
   System.out.println("Different ages: "+globalMap.get("number"));
   System.out.println("Lowest age: "+((Integer[])globalMap.get("tMemorizeRows_1_age"))[0]);
   ```

   The method `globalMap.get()` is used by the **tJava** to retrieve the array values. Note that here, the **tJava** is used outside the subjob **tMemorizeRows** so the variable format should be `ComponentName_ColumnName`, which is different from the variable format used by a component placed in the same flow.

6. In the **Code** area, enter the following code to display the number of occurrences of different ages and the lowest age within the customers on the console:

   ```java
   System.out.println("Different ages: "+globalMap.get("number"));
   System.out.println("Lowest age: "+((Integer[])globalMap.get("tMemorizeRows_1_age"))[0]);
   ```

   Saving and executing the Job

   **Procedure**

1. Press **Ctrl+S** to save your Job.
2. Press **F6**, or click **Run** on the **Run** console to execute the Job.

   ![Console Output](image2)

   **Results**

   In the console, you can read that there are four different ages and that the lowest age is 27 within the group of customers.
tMemSQLClose

Closes the transaction committed in the MemSQL database.

**tMemSQLClose Standard properties**

These properties are used to configure tMemSQLClose running in the Standard Job framework.

The Standard tMemSQLClose component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tMemSQLConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see **Talend Studio User Guide**. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see **Talend Studio User Guide**. |

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see **Talend Studio User Guide**. |

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see **Talend Studio User Guide**. |
### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with the other MemSQL components, especially with tMemSQLConnection.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

### Related scenario

For a related scenario, see Scenario: Writing data to and reading data from a MemSQL database table on page 2230.
tMemSQLConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

**tMemSQLConnection Standard properties**

These properties are used to configure tMemSQLConnection running in the Standard Job framework. The Standard tMemSQLConnection component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>:</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong>:</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Host</td>
<td>Enter the location of the database server to be used.</td>
</tr>
<tr>
<td>Port</td>
<td>Enter the listening port number of the database server to be used.</td>
</tr>
<tr>
<td>Database</td>
<td>Enter the name of the database to be used.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td>Use or register a shared DB Connection</td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the Shared DB Connection Name field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child. This option is incompatible with the Use dynamic job and Use an independent process to run subjob options of the tRunJob component. Using a shared connection.</td>
</tr>
</tbody>
</table>
together with a tRunJob component with either of these two options enabled will cause your Job to fail.

Advanced settings

<table>
<thead>
<tr>
<th><strong>Additional JDBC parameters</strong></th>
<th>Specify additional connection properties for the DB connection you are creating.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

Global Variables

**Global Variables**

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

**Usage rule**

This component is more commonly used with other MemSQL components.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Related scenario

For a related scenario, see Scenario: Writing data to and reading data from a MemSQL database table on page 2230.
tMemSQLInput

Executes a DB query with a strictly defined order which must correspond to the schema definition. tMemSQLInput connects to a given database and extracts fields based on a user-defined query. Then it passes the selected data to the next component via a Main row link.

**tMemSQLInput Standard properties**

These properties are used to configure tMemSQLInput running in the Standard Job framework.
The Standard tMemSQLInput component belongs to the Big Data and the Databases families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td>1.</td>
<td>In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td>2.</td>
<td>In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
</tbody>
</table>
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Enter the location of the server of the MemSQL database to be used.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Enter the listening port number of the server of the MemSQL database to be used.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Enter the name of the MemSQL database to be used.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>Enter the authentication data used to connect to the MemSQL database to be used.</td>
</tr>
</tbody>
</table>

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Table Name**

Name of the table to be read.

**Query type** and **Query**

Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

### Advanced settings

**Additional JDBC parameters**

Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the *Basic settings*. 
### Note:
When you need to handle data of the time-stamp type `0000-00-00 00:00:00` using this component, set the parameter as:

`noDateTimeStringSync=true&zeroDateTimeBehavior=convertToNull`.

<table>
<thead>
<tr>
<th>Trim all the String/Char columns</th>
<th>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trim column</td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td>Note:</td>
<td>Clear Trim all the String/Char columns to enable Trim columns in this field.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Variables</td>
<td><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>QUERY</strong>: the query statement being processed. This is a Flow variable and it returns a string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <strong>Talend Studio User Guide</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible SQL queries for MemSQL databases.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the <code>[+ ]</code> button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example,</td>
</tr>
</tbody>
</table>
when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |

---

**Scenario: Writing data to and reading data from a MemSQL database table**

This scenario applies only to a Talend solution with Big Data.

In this scenario a Job is built to write a table to a MemSQL database, and then retrieve the data from the table.
Dragging and dropping components and linking them together

About this task

![Diagram showing component connections]

Procedure

1. Drop a `tMemSQLConnection`, a `tRowGenerator`, a `tMemSQLInput`, a `tMemSQLOutput`, a `tMemSQLClose`, and a `tLogRow` from the Palette onto the workspace.
2. Link the `tMemSQLConnection` to the `tRowGenerator` using a `Row > OnSubjobOk` connection.
3. Link the `tRowGenerator` to the `tMemSQLOutput` using a `Row > Main` connection.
4. Link the `tRowGenerator` to the `tMemSQLInput` using a `Row > OnSubjobOk` connection.
5. Link the `tMemSQLInput` to the `tLogRow` using a `Row > Main` connection.
6. Link the `tMemSQLInput` to the `tMemSQLClose` using a `Row > OnSubjobOk` connection.

Configuring the components

Opening and closing connection to the database

Procedure

1. Double-click the `tMemSQLConnection` component to open its Basic settings in the Component tab.
2. In the **Property Type** list, select **Built-In**. Specify the connection details in the relevant fields, including the host name and listening port number of the database server, the user name and password for your database authentication, and the database name.

3. Double-click **tMemSQLClose** to set its **Basic settings** in the **Component** tab.

4. The **Component List** is filled in automatically with **tMemSQLConnection_1**.

**Writing data in a database table**

**Procedure**

1. Double-click the **tRowGenerator** component to open the Row Generator editor.
2. Add three columns as shown below to generate 10 data rows:
   - **id**, Integer type, using **Numeric.sequence** from the **Function** list to generate sequence numbers.
   - **firstName**, String type, using **TalendDataGenerator.getFirstName** from the **Function** list to generate random first names.
   - **lastName**, String type, using **TalendDataGenerator.getLastName** from the **Function** list to generate random family names.

3. In the **Basic settings** view of **tRowGenerator** in the **Component** tab, click the **Edit Schema** button to open the schema editor. Set the length of the **id**, **firstName**, **lastName** column to 4, 10, and 10.
When done, click **OK** to close the editor and click **Yes** when prompted to propagate the schema to the next component.

For more information about how to configure the `tRowGenerator` component, see `tRowGenerator` on page 3281.

4. Double-click `tMemSQLOutput` to open its **Basic Settings** view in the **Component** tab.

5. Select the **Use Existing Connection** check box. The **Component List** is filled in automatically with `tMemSQLConnection_1`.

   Fill the **Table** field with the database table, *customers* in this example.

   Select **Drop table if exists and create** in the **Action on table** list, and **Insert** in the **Action on table** list.

**Reading data from a database table**

**Procedure**

1. Double-click `tMemSQLInput` to open its **Basic Settings** view in the **Component** tab.
2. Select the **Use Existing Connection** check box. The **Component List** is filled in automatically with **tMemSQLConnection_1**.

3. Set the **Schema** as **Built-in** and click **Edit schema** to define the data structure of the database table to read data from.

   - Click the [+•] button to add the rows that you will use to define the schema, three columns in this example: **id**, **firstName**, and **lastName**.
   - Under **Column**, click in the fields to enter the corresponding column names.
   - Click the field under **Type** to define the type of data.
   - Click **OK** to close the schema editor.

4. Fill the **Table** field with the database table, **customers** in this example.
   - Alternatively, click the [...] button next to the **Table Name** field to select the database table of interest. A dialog box opens showing a tree diagram of all the tables in the selected database.
   - Select **customers** and click **OK** to close the dialog box.
5. Set the **Query Type** as **Built-In**. Click the **Guess Query** button. The **Query** box is filled in automatically to retrieve all columns from the selected table.

6. Double-click **tLogRow** to set its **Basic settings** in the **Component** tab.

7. In the **Mode** area, select **Table (print values in cells of a table)** for a better display of the results.

8. Save the Job.

**Executing the Job**

Press **F6** to run the Job and see the results in the console.
As shown above, the data with the desired column names is retrieved from the database table.

<table>
<thead>
<tr>
<th>id</th>
<th>firstName</th>
<th>lastName</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Thomas</td>
<td>Quincy</td>
</tr>
<tr>
<td>2</td>
<td>Jimmy</td>
<td>Jefferson</td>
</tr>
<tr>
<td>3</td>
<td>Jimmy</td>
<td>Madison</td>
</tr>
<tr>
<td>4</td>
<td>Ronald</td>
<td>Ford</td>
</tr>
<tr>
<td>5</td>
<td>Thomas</td>
<td>Buchanan</td>
</tr>
<tr>
<td>6</td>
<td>William</td>
<td>Washington</td>
</tr>
<tr>
<td>7</td>
<td>Thomas</td>
<td>Harrison</td>
</tr>
<tr>
<td>8</td>
<td>Bill</td>
<td>Truman</td>
</tr>
<tr>
<td>9</td>
<td>Lyndon</td>
<td>Roosevelt</td>
</tr>
<tr>
<td>10</td>
<td>Dwight</td>
<td>Clinton</td>
</tr>
</tbody>
</table>
tMemSQLOutput

Reads data incoming from the preceding component in the Job and executes the action defined on a given MemSQL table and/or on the data contained in the table.

tMemSQLOutput connects to a given MemSQL database and writes, updates, makes changes or suppresses entries in that database.

**tMemSQLOutput Standard properties**

These properties are used to configure tMemSQLOutput running in the Standard Job framework. The Standard tMemSQLOutput component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. |

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](#).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Enter the location of the server of the MemSQL database to be used.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Enter the listening port number of the database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Enter the name of the MemSQL database to be used.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Enter the user authentication data for connecting to the database to be used.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Enter the name of the table to be written. Note that only one table can be written at a time</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td><strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create a table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table if not exists</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop a table if exists and create</strong>: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear a table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td><strong>Action on data</strong></td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, the job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Make changes to existing entries.</td>
</tr>
<tr>
<td></td>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td><strong>Update or insert</strong>: Update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Remove entries corresponding to the input flow.</td>
</tr>
</tbody>
</table>

**Warning:**

You must specify at least one column as a primary key on which the **Update** and **Delete** operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the update and delete operations. To do that: Select the **Use field options** check box and then in the **Key in update** column, select the check boxes next to the column name on which you want to base the update operations.
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

### Built-In

You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

### Repository

You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.

### Die on error

This check box is clear by default. This allows you to skip the row in error and complete the process for error-free rows.

### Advanced settings

Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the **Basic settings**.

**Note:**

You can press **Ctrl+Space** to access a list of predefined global variables.
## Use batch size

Select this check box to activate the batch mode for data processing. In the **Batch Size** field that appears when this check box is selected, you can type in the number you need to define the batch size to be processed.

## Additional Columns

This option is not available if you have just created the database table (even if you delete it beforehand). This option allows you to call SQL functions to perform actions on columns, provided that these are not insert, update or delete actions, or actions that require pre-processing.

**Name:** Type in the name of the schema column to be altered or inserted.

**SQL expression:** Type in the SQL statement to be executed in order to alter or insert the data in the corresponding column.

**Position:** Select **Before**, **Replace** or **After**, depending on the action to be performed on the reference column.

**Reference column:** Type in a reference column that **tMemSQLOutput** can use to locate or replace the new column, or the column to be modified.

## Use field options

Select this check box to customize a request, particularly if multiple actions are being carried out on the data.

## Enable debug mode

Select this check box to display each step involved in the process of writing data in the database.

## tStatCatcher Statistics

Select this check box to collect log data at the component level.

## Global Variables

**Global Variables**

**NB_LINE:** the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

**NB_LINE_UPDATED:** the number of rows updated. This is an After variable and it returns an integer.

**NB_LINE_INSERTED:** the number of rows inserted. This is an After variable and it returns an integer.

**NB_LINE_DELETED:** the number of rows deleted. This is an After variable and it returns an integer.

**NB_LINE_REJECTED:** the number of rows rejected. This is an After variable and it returns an integer.

**ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

### Usage

**Usage rule**

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a MemSQL database.

**Dynamic settings**

Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### Related scenario

For a related scenario, see *Scenario: Writing data to and reading data from a MemSQL database table* on page 2230.
tMemSQLRow

Acts on the actual database structure or on the data (although without handling data).

tMemSQLRow executes the MemSQL query stated in the specified database, depending on the nature of the query and the database. The row suffix means the component implements a flow in the job design although it does not provide output.

The SQLBuilder tool helps you write easily your SQL statements.

**tMemSQLRow Standard properties**

These properties are used to configure tMemSQLRow running in the Standard Job framework.

The Standard tMemSQLRow component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
</tbody>
</table>
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Enter the location of the server of the MemSQL database to be used.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Enter the listening port number of the database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Enter the name of the MemSQL database to be used.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Enter the user authentication data for connecting to the database to be used.</td>
</tr>
<tr>
<td><strong>Table Name</strong></td>
<td>Name of the table to be processed.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>. When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.
<table>
<thead>
<tr>
<th><strong>Query type</strong></th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: Fill in manually the query statement or build it graphically using SQLBuilder</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the relevant query stored in the Repository. The <strong>Query</strong> field gets accordingly filled in.</td>
<td></td>
</tr>
<tr>
<td><strong>Guess Query</strong></td>
<td>Click the <strong>Guess Query</strong> button to generate the query which corresponds to your table schema in the <strong>Query</strong> field.</td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is clear by default. This allows you to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>

### Advanced settings

| **Additional JDBC parameters** | Specify additional connection properties for the database connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the **Basic settings**. |
| **Propagate QUERY’s record set** | Select this check box to insert the result of the query in a COLUMN of the current flow. Select this column from the **use column** list.  

**Note:**  
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s record set should be set to the type of **Object** and this component is usually followed by **tParseRecordSet**. |
| **Use PreparedStatement** | Select this check box if you want to query the database using a PreparedStatement. In the **Set PreparedStatement Parameter** table, define the parameters represented by “?“ in the SQL instruction of the **Query** field in the **Basic Settings** tab.  

**Parameter Index**: Enter the parameter position in the SQL instruction.  
**Parameter Type**: Enter the parameter type.  
**Parameter Value**: Enter the parameter value.  

**Note:**  
This option is very useful if you need to execute the same query several times. Performance levels are increased |
| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | QUERY: the query statement being processed. This is a Flow variable and it returns a string.  
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

**Usage**

| Usage rule | This component offers the flexibility of the database query and covers all possible SQL queries. |

| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  
The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.  
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

**Limitation**

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add |
all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tMicrosoftCrmInput

Extracts data from a Microsoft CRM database based on conditions set on specific columns.

tMicrosoftCrmInput Standard properties

These properties are used to configure tMicrosoftCrmInput running in the Standard Job framework. The Standard tMicrosoftCrmInput component belongs to the Business family. The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Authentication Type</th>
<th>Select the authentication type that corresponds to your specific CRM deployment model.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>ON_PREMISE</strong>: the CRM is installed in-house on the customer’s own servers, and licensing is purchased along with any necessary infrastructure.</td>
</tr>
<tr>
<td></td>
<td>• <strong>ONLINE</strong>: the customer accesses the CRM as part of a cloud-based subscription service, hosted by Microsoft in their data centers.</td>
</tr>
</tbody>
</table>

| CRM Version | Select the version of the ON-PREMISE CRM database, CRM 2011, CRM 2015, or CRM 2016. ON-PREMISE 2016 is implemented using Apache Olingo OData 4.0 libraries. For more information, see http://olingo.apache.org/doc/odata4/index.html. |


| Registered application type | Select the type of your application, either **Native App** or **Web App with delegated permissions**. This property is available for Microsoft CRM ONLINE 2016. |

| Microsoft WebService URL | The Web Service URL of an on-premise Microsoft CRM server. This property is available for Microsoft CRM ON-PREMISE 2011. |

| OrganizeName | The name of the organization that needs to access the Microsoft CRM database. This property is available for Microsoft CRM ON-PREMISE 2011, ONLINE 2007, and ONLINE 2011. |

<p>| Username and Password | The user authentication data required to access the Microsoft CRM database. |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>The domain name of the server on which the Microsoft CRM database is hosted.</td>
<td>Microsoft CRM ON-PREMISE 2011 and 2016.</td>
</tr>
<tr>
<td>Port</td>
<td>The listening port number of the Microsoft CRM database server.</td>
<td>Microsoft CRM ON-PREMISE 2011.</td>
</tr>
<tr>
<td>Discovery service endpoint</td>
<td>The CRM online discovery service endpoint.</td>
<td>Microsoft CRM ONLINE 2011.</td>
</tr>
<tr>
<td>Service Root URL</td>
<td>The root URL of the service to be connected.</td>
<td>Microsoft CRM ON-PREMISE 2016 and ONLINE 2016.</td>
</tr>
<tr>
<td>Application ID</td>
<td>The unique client identifier that Azure AD assigns to your application when registering the application in the Azure portal. For more information, see Register a Dynamics 365 app with Azure Active Directory.</td>
<td>Microsoft CRM ONLINE 2016.</td>
</tr>
<tr>
<td>Application secret</td>
<td>The access key for your web application. For more information about how to get its value, see Web App with delegated permissions is selected from the Registered application type drop-down list.</td>
<td>This property is available when Web App with delegated permissions is selected from the Registered application type drop-down list.</td>
</tr>
</tbody>
</table>
| **OAuth authorization endpoint** | The OAuth authorization endpoint to be used for the OAuth discovery. For more information, see [OAuth authorization endpoints](#).  
This property is available for Microsoft CRM ONLINE 2016. |
| **Time out (second)** | The number of seconds to wait for a reply message from the CRM server before closing the connection to it. |
| **Entity/EntitySet** | Select the relevant entity/entity set from the list or select CustomEntity/CustomEntitySet from the list, and then in the **Custom Entity Name** field displayed, specify the entity/entity set name and define the schema for the custom entity/entity set. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.  
- **Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](#).  
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](#).  
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.  
The schema of this component is related to the specified entity. |
| **Custom filter** | Select this check box and in the field displayed, specify the search criteria that uses the standard OData query functions such as `contains`, `endswith`, and `startswith`. For more information about the supported query functions, see the description of Standard query functions on [Query Data using the Web API](#).  
This property is available for Microsoft CRM ONLINE 2016 and ONLINE 2016. |
### Logical operator used to combine conditions

Select the logical operator used to combine the conditions defined in the **Conditions** table.

- **And**: the record will be included in the result set when all of the conditions are met.
- **Or**: the record will be included in the result set when any of the conditions is met.

### Conditions

Click the [*] button to add as many conditions as needed and specify the following parameters for each condition:

- **Input column**: Click in the cell and select the column of the input schema on which the condition is to be set.
- **Operator**: Click in the cell and select the operator used to compare the value for the input column with the value defined in the **Value** column. The values for this parameter are: Equal, NotEqual, GreaterThan, LessThan, GreaterEqual, LessEqual, and Like.
- **Value**: Type in the value between quotes if needed.

### Order by

Select this check box and in the table displayed, specify the column(s) based on which the retrieved data will be sorted in ascending or descending order. Multiple sort columns can be specified and the hierarchy of the sorting is based on the order of the columns specified. That is to say, the retrieved data will be sorted by the first column and then that ordered list is sorted by the second column, and so on.

This property is available for Microsoft CRM ON-PREMISE 2016 and ONLINE 2016.

### Advanced settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reuse Http Client</strong></td>
<td>Select this check box to retain the current connection or clear it to release the connection. This property is not available for Microsoft CRM ON-PREMISE 2015.</td>
</tr>
<tr>
<td><strong>Max number of reconnection attempts</strong></td>
<td>The maximum number of reconnect attempts to the Microsoft CRM database after the token is expired. If the value is set to 0 or less than 0, no reconnect attempt will be made. This property is available for Microsoft CRM ON-PREMISE 2016, ONLINE 2011, and ONLINE 2016.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>
Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage rule</td>
<td>This component is usually used as a start component of a Job or Subjob and it always needs an output link.</td>
</tr>
</tbody>
</table>

Limitation

<table>
<thead>
<tr>
<th>Limitation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>tMicrosoftCrmInput (both ONLINE 2016 and ON-PREMISE 2016) doesn’t support retrieving data of the write-only properties. The corresponding schema columns for the write-only properties have been removed from the predefined schema.</td>
</tr>
</tbody>
</table>

Scenario: Writing data in a Microsoft CRM database and putting conditions on columns to extract specified rows

This scenario describes a four-component Job which aims at writing the data included in a delimited input file in a custom entity in a Microsoft CRM database. It then extracts specified rows to an output file using the conditions set on certain input columns.

⚠️ Warning:
If you want to write in a CustomEntity in Microsoft CRM database, make sure to name the columns in accordance with the naming rule set by Microsoft, that is "name_columnname" all in lower case.

Setting up the Job

Procedure

1. Drop the following components from the Palette to the design workspace: tFileInputDelimited, tFileOutputDelimited, tMicrosoftCrmInput, and tMicrosoftCrmOutput.
2. Connect `tFileInputDelimited` to `tMicrosoftCrmOutput` using a Row > Main connection.
3. Connect `tMicrosoftCrmInput` to `tFileOutputDelimited` using a Row > Main connection.
4. Connect `tFileInputDelimited` to `tMicrosoftCrmInput` using Trigger > OnSubjobOk connection.

**Configuring tFileInputDelimited**

**Procedure**

1. Double-click `tFileInputDelimited` to display its Basic settings view and define its properties.

   ![Basic settings view](image)

   - **Property Type**: Set to Repository if you have stored the input file properties centrally in the Metadata node in the Repository tree view. Otherwise, select Built-In and fill the fields that follow manually. In this example, property is set to Built-In.
   - **File name/Stream**: Browse to the delimited file that holds the input data. The input file in this example contains the following columns: `new_id`, `new_status`, `new_firstname`, `new_email`, `new_city`, `new_initial`, `new_zipcode`.

   ```
   1; married; Paul; jnewman@comp.com; New York; P; M; 55677
   2; single; Raul; pnewman@comp.com; New York; R; L; 55677
   3; single; Mary; mnewman@comp.com; Chicago; M; B; 66898
   4; married; John; Jnewman@comp.com; Chicago, J; M; 66898
   5; single; Martin; mnewman@comp.com; Sunnyvale, M; P; 33662
   6; married; Janet; jnewman@comp.com; Sunnyvale, J; P; 33662
   7; married; Harry; hnewman@comp.com; New York; H; M; 55677
   8; married; Jerry; jnewman@comp.com; New York; J; M; 55677
   9; married; Alice; anewman@comp.com; New York; A; M; 55677
   10; single; Jack; jnewman@comp.com; New York; J; M; 55677
   ```

2. Set the Property Type to Repository if you have stored the input file properties centrally in the Metadata node in the Repository tree view. Otherwise, select Built-In and fill the fields that follow manually. In this example, property is set to Built-In.
3. Click the three-dot button next to the File name/Stream field and browse to the delimited file that holds the input data. The input file in this example contains the following columns: `new_id`, `new_status`, `new_firstname`, `new_email`, `new_city`, `new_initial` and `new_zipcode`.

4. In the Basic settings view, define the Row Separator allowing to identify the end of a row. Then define the Field Separator used to delimit fields in a row.
5. If needed, define the header, footer and limit number of processed rows in the corresponding fields. In this example, the header, footer and limits are not set.
6. Click Edit schema to open a dialog box where you can define the input schema you want to write in Microsoft CRM database.
7. Click **OK** to close the dialog box.

**Configuring tMicrosoftCrmOutput**

**Procedure**

1. Double-click **tMicrosoftCrmOutput** to display the component **Basic settings** view and define its properties.

2. Enter the Microsoft Web Service URL as well as the user name and password in the corresponding fields.

3. In the **OrganizeName** field, enter the name that is given the right to access the Microsoft CRM database.

4. In the **Domain** field, enter the domain name of the server on which Microsoft CRM is hosted, and then enter the host IP address and the listening port number in the corresponding fields.

5. In the **Action** list, select the operation you want to carry on. In this example, we want to insert data in a custom entity in Microsoft Crm.

6. In the **Time out** field, set the amount of time (in seconds) after which the Job will time out.

7. In the **Entity** list, select one among those offered. In this example, **CustomEntity** is selected.

**Note:**
If `CustomEntity` is selected, a **Custom Entity Name** field displays where you need to enter a name for the custom entity.

The **Schema** is then automatically set according to the entity selected. If needed, click **Edit schema** to display a dialog box where you can modify this schema and remove the columns that you do not need in the output.

8. Click **Sync columns** to retrieve the schema from the preceding component.

---

### Configuring tMicrosoftCrmInput

**Procedure**

1. Double-click **tMicrosoftCrmInput** to display the component **Basic settings** view and define its properties.

2. Set the **Property Type** to **Repository** if you have stored the input file properties centrally in the **Metadata** node in the **Repository** tree view. Otherwise, select **Built-In** and fill the fields that follow manually. In this example, property is set to **Built-In**.
3. Enter the Microsoft Web Service URL as well as the user name and password in the corresponding fields and enter the name that is given the right to access the Microsoft CRM database in the OrganizeName field.

4. In the Domain field, enter the domain name of the server on which Microsoft CRM is hosted, and then enter the host IP address and the listening port number in the corresponding fields.

5. In the Time out field, set the amount of time (in seconds) after which the Job will time out.

6. In the Entity list, select the one among those offered you want to connect to. In this example, CustomEntity is selected.

7. The Schema is then automatically set according to the entity selected. But you can modify it according to your needs. In this example, you should set the schema manually since you want to access a custom entity. Copy the seven-column schema from tMicrosoftCrmOutput and paste it in the schema dialog box in tMicrosoftCrmInput.

8. Click OK to close the dialog box. You will be prompted to propagate changes. Click Yes in the popup message.

9. In the Basic settings view, select And or Or as the logical operator you want to use to combine the conditions you set on the input columns. In this example, we want to set two conditions on two different input columns and we use And as the logical operator.

10. In the Condition area, click the plus button to add as many lines as needed and then click in each line in the Input column list and select the column you want to set condition on. In this example, we want to set conditions on two columns, new_city and new_id. We want to extract all customer rows whose city is equal to “New York” and whose id is greater than 2.

11. Click in each line in the Operator list and select the operator to bind the input column with its value, in this example Equal is selected for new_city and Greater Than for new_id.

12. Click in each line in the Value list and set the column value, New York for new_city and 2 for new_id in this example. You can use a fixed or a context value in this field.
Configuring tFileOutputDelimited

Procedure

1. Double-click tFileOutputDelimited to display the component Basic settings view and define its properties.

2. Set Property Type to Built-In and then click the [...] button next to the File Name field and browse to the output file.

3. Set row and field separators in the corresponding fields.

4. Select the Append check box if you want to add the new rows at the end of the records.

5. Select the Include Header check box if the output file includes a header.

6. Click Sync columns to retrieve the schema from the preceding component.

Executing the Job

Procedure

Save the Job and press F6 to execute it.

Results

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7;married;Harry;<a href="mailto:hnewman@comp.com">hnewman@comp.com</a>;New York;H.M;55677</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8;married;Jerry;<a href="mailto:jnewman@comp.com">jnewman@comp.com</a>;New York;J.M;55677</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>9;married;Alice;<a href="mailto:anewman@comp.com">anewman@comp.com</a>;New York;A.M;55677</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10;single;Jack;<a href="mailto:jnewman@comp.com">jnewman@comp.com</a>;New York;J.M;55677</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Only customers who live in New York city and those whose "id" is greater than 2 are listed in the output file you stored locally.
tMicrosoftCrmOutput

Writes data into a Microsoft CRM database.

**tMicrosoftCrmOutput Standard properties**

These properties are used to configure tMicrosoftCrmOutput running in the Standard Job framework.

The Standard tMicrosoftCrmOutput component belongs to the Business family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Authentication Type</th>
<th>Select the authentication type that corresponds to your specific CRM deployment model.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>ON PREMISE</strong>: the CRM is installed in-house on the customer’s own servers, and licensing is purchased along with any necessary infrastructure.</td>
</tr>
<tr>
<td></td>
<td>• <strong>ONLINE</strong>: the customer accesses the CRM as part of a cloud-based subscription service, hosted by Microsoft in their data centers.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CRM Version</th>
<th>Select the version of the ON-PREMISE CRM database, CRM 2011, CRM 2015, or CRM 2016.</th>
</tr>
</thead>
</table>

|---------------------|-------------------------------------------------------------------------------------|

<table>
<thead>
<tr>
<th>Registered application type</th>
<th>Select the type of your application, either Native App or Web App with delegated permissions.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This property is available for Microsoft CRM ONLINE 2016.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Microsoft WebService URL</th>
<th>The Web Service URL of an on-premise Microsoft CRM server.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This property is available for Microsoft CRM ON-PREMISE 2011.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OrganizeName</th>
<th>The name of the organization that needs to access the Microsoft CRM database.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This property is available for Microsoft CRM ON-PREMISE 2011, ONLINE 2007, and ONLINE 2011.</td>
</tr>
</tbody>
</table>

| Username and Password      | The user authentication data required to access the Microsoft CRM database. |
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Available in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>The domain name of the server on which the Microsoft CRM database is hosted.</td>
<td>Microsoft CRM ON-PREMISE 2011 and 2016</td>
</tr>
<tr>
<td>Host</td>
<td>The IP address of the Microsoft CRM database server.</td>
<td>Microsoft CRM ON-PREMISE 2011, ON-PREMISE 2016, and ONLINE 2007</td>
</tr>
<tr>
<td>Port</td>
<td>The listening port number of the Microsoft CRM database server.</td>
<td>Microsoft CRM ON-PREMISE 2011</td>
</tr>
<tr>
<td>Organization Wsdl URL</td>
<td>The WSDL URL of the Organization Service.</td>
<td>Microsoft CRM ON-PREMISE 2015</td>
</tr>
<tr>
<td>Trust Certificate</td>
<td>The path to the Trust Certificate file.</td>
<td>Microsoft CRM ON-PREMISE 2015</td>
</tr>
<tr>
<td>Discovery service endpoint</td>
<td>The CRM online discovery service endpoint.</td>
<td>Microsoft CRM ONLINE 2011</td>
</tr>
<tr>
<td>Service Root URL</td>
<td>The root URL of the service to be connected.</td>
<td>Microsoft CRM ON-PREMISE 2016 and ONLINE 2016</td>
</tr>
<tr>
<td>Application ID</td>
<td>The unique client identifier that Azure AD assigns to your application when registering the application in the Azure portal. For more information, see Register a Dynamics 365 app with Azure Active Directory.</td>
<td>Microsoft CRM ONLINE 2016</td>
</tr>
<tr>
<td>Application secret</td>
<td>The access key for your web application. For more information about how to get its value, see .</td>
<td>When Web App with delegated permissions is selected from the Registered application type drop-down list.</td>
</tr>
</tbody>
</table>
### OAuth authorization endpoint
The OAuth authorization endpoint to be used for the OAuth discovery. For more information, see OAuth authorization endpoints.
This property is available for Microsoft CRM ONLINE 2016.

### Action
Select the action you want to perform on the CRM data, **insert**, **update**, or **delete**.
The CRM data could not be updated with NULL value for Microsoft CRM ONLINE 2016 because of a limitation from Apache. For more information about the limitation, see [https://issues.apache.org/jira/browse/OLINGO-1114](https://issues.apache.org/jira/browse/OLINGO-1114).

### Time out (second)
The number of seconds to wait for a reply message from the CRM server before closing the connection to it.

### Entity/EntitySet
Select the relevant entity/entity set from the list or select CustomEntity/CustomEntitySet from the list, and then in the **Custom Entity Name** field displayed, specify the entity/entity set name and define the schema for the custom entity/entity set.

### Schema and Edit schema
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](https://www.talend.com/).
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](https://www.talend.com/).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

Click **Sync columns** to retrieve the schema from the previous component connected in the Job.
The schema of this component is related to the specified entity.

### Lookup Type Mapping
Complete this table to map the lookup property with its related entity type or entity set value.
- **Input column**: the schema column name of the lookup property.
- **Type**: the related entity type for the corresponding lookup property. It is not available for ON-PREMISE 2016 and ONLINE 2016.
- **Reference entityset**: the name of the related entity set for the corresponding lookup property. It is available only for ON-PREMISE 2016 and ONLINE 2016. Refer to Web API EntityType Reference for more information about the entity set of the lookup property for each entity.

This property is not available for Microsoft CRM ON-PREMISE 2015.

### Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reuse HttpClient</strong></td>
<td>Select this check box to retain the current connection or clear it to release the connection. This property is not available for Microsoft CRM ON-PREMISE 2015.</td>
</tr>
<tr>
<td><strong>Transform empty lookup string values to NULL</strong></td>
<td>Select this check box and the empty lookup string will be written into the Microsoft CRM database as the NULL value. This property is available for Microsoft CRM ON-PREMISE 2016, ONLINE 2011, and ONLINE 2016.</td>
</tr>
<tr>
<td><strong>Ignore Null</strong></td>
<td>Select this check box to ignore NULL values in the input data.</td>
</tr>
<tr>
<td><strong>Max number of reconnection attempts</strong></td>
<td>The maximum number of reconnect attempts to the Microsoft CRM database after the token is expired. If the value is set to 0 or less than 0, no reconnect attempt will be made. This property is available for Microsoft CRM ON-PREMISE 2016, ONLINE 2011, and ONLINE 2016.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong></td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>
Usage

| Usage rule   | This component is usually used as an end component of a Job or Subjob and it always needs an input link. |

Related Scenario

For a related use case, see *Scenario: Writing data in a Microsoft CRM database and putting conditions on columns to extract specified rows* on page 2252.
**tMicrosoftMQInput**

Retrieves the first message in a given Microsoft message queue (only support String).

*tMicrosoftMQInput* fetches messages one by one based on the ID sequence of these messages from the Microsoft message queue. Each execution retrieves only one message.

**tMicrosoftMQInput Standard properties**

These properties are used to configure *tMicrosoftMQInput* running in the Standard Job framework.

The Standard *tMicrosoftMQInput* component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property type</strong></td>
<td>Always <em>Built-in</em>. Its schema has only one read-only column of string type: <em>message</em>.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Type in the Host name or IP address of the host server.</td>
</tr>
<tr>
<td><strong>Queue</strong></td>
<td>Enter the queue name you want to retrieve messages from.</td>
</tr>
</tbody>
</table>
| **Queue type** | Select a Microsoft message queue type from the drop-down list.  
- **Private**: a private queue, which is registered on the local machine.  
- **Public**: a public queue, which is registered in the directory service.  
| **Die on error** | This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |
### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <em>Die on error</em> check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press <code>Ctrl + Space</code> to access the variable list and choose the variable to use from it. For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>This component is generally used as a start component of a Job or Subjob. It must be linked to an output component.</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connections</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outgoing links (from this component to another):</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Row:</strong> Main, Iterate</td>
<td></td>
</tr>
<tr>
<td><strong>Trigger:</strong> Run if; On Subjob Ok, On Subjob Error, On Component Ok; On Component Error.</td>
<td></td>
</tr>
<tr>
<td><strong>Incoming links (from one component to this one):</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Row:</strong> Iterate;</td>
<td></td>
</tr>
<tr>
<td><strong>Trigger:</strong> Run if, On Subjob Ok, On Subjob Error, On Component Ok, On Component Error.</td>
<td></td>
</tr>
<tr>
<td>For further information regarding connections, see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Limitation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>This component supports only String type. Also, it only works with the Windows systems.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>This component requires installation of its related jar files.</strong></td>
<td></td>
</tr>
</tbody>
</table>

### Scenario: Writing and fetching queuing messages from Microsoft message queue

This scenario is made of two Jobs. The first Job posts messages on a Microsoft message queue and the second Job fetches the message from the server.

### Posting messages on a Microsoft message queue

In the first Job, a string message is created using a *tRowGenerator* and put on a Microsoft message queue using a *tMicrosoftMQOutput*. An intermediary *tLogRow* component displays the flow being passed.
Dropping and linking components

Procedure

1. Drop the three components required for the first Job from the Palette onto the design workspace.
2. Connect the components using a Row > Main link.

![Diagram showing connections between components](image)

Configuring the components

Procedure

1. Double-click tRowGenerator to open its editor.

<table>
<thead>
<tr>
<th>Column</th>
<th>Key</th>
<th>Type</th>
<th>✓</th>
<th>N., Le...</th>
<th>Pr...</th>
<th>D...</th>
<th>C...</th>
<th>Functions</th>
<th>Environ...</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td></td>
<td>Integer</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>random</td>
<td>min value..</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>getFirstName</td>
<td>getUsCity</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Click the plus button to add three rows into the schema table.
3. In the **Column** column, type in a new name for each row to rename it. Here, we type in *ID*, *Name* and *Address*.
4. In the **Type** column, select **Integer** for the *ID* row from the drop-down list and leave the other rows as **String**.
5. In the **Functions** column, select **random** for the *ID* row, **getFirstName** for the *Name* row and **getUsCity** for the *Address* row.
6. In the **Number of Rows for RowGenerator** field on the right end of the toolbar, type in 12 to limit the number of rows to be generated. Then, Click **Ok** to validate this editing.

⚠️ Note:

In real case, you may use an input component to load the data of your interest, instead of the tRowGenerator component.

7. Double click the tMicrosoftMQOutput component to open its Component view.
8. In the **Host** field, type in the host address. In this example, it is *localhost*.

9. In the **Queue** field, type in the queue name you want to write message in. In this example, name it *AddressQueue*.

10. In **Message column (String Type)** field, select *Address* from the drop-down list to determine the message body to be written.

### Fetching the message

#### Procedure

1. Press **Ctrl+S** to save your Job.

2. Press **F6** or click **Run** on the **Run** tab to execute the Job.

```
Starting job tMicrosoftMQ at 14:03 18/11/2010.
[statistics] connecting to socket on port 3852
[statistics] connected
Queue open failure: Cannot open queue. (hr=MQ_ERROR_QUEUE_NOT_FOUND)
OpenQueueWithAccess (DIRECT-OS:\localhost\private$\AddressQueue)
open. fnname(DIRECT-OS:\localhost\private$\AddressQueue) accessmode(1) sharemode(0)

.................. attempting to create queue with
name='\private$\AddressQueue', label='Created by
corg.talend.mqm.MemqUtil.java'
OpenQueueWithAccess (DIRECT-OS:\private$\AddressQueue)
open. fnname(DIRECT-OS:\private$\AddressQueue) accessmode(1) sharemode(0)
open. fnname(DIRECT-OS:\private$\AddressQueue) accessmode(2) sharemode(0)
7/Chicago|Atlanta
8/Andrew|Saint Paul
7/Benjamin|Oklahoma City
3/Abraham|Providence
4/Rutherford|Lincoln
6/Chester|Little Rock
11/Richard|Lincoln
8/Bill|Boston
9/Herbert|Jackson
10/James|Honolulu
5/Jimmy|Montpelier
9/Vooodrow|Nashville
[statistics] disconnected
Job tMicrosoftMQ ended at 14:03 18/11/2010 [exit code=0]
```

You can see that this queue has been created automatically and that the messages have been written.
**Fetching the first queuing message from the message queue**

Now set the second Job in order to fetch the first queuing message from the message queue.

**Arranging the flow of the message**

**Procedure**

1. Drop **tMicrosoftMQInput** and **tLogRow** from the Palette to the design workspace.
2. Connect these two components using a **Row > Main** link.

![Diagram of the flow of the message with **tMicrosoftMQInput** and **tLogRow** connected](image)

**Configuring how the message is processed**

**Procedure**

1. Double-click the **tMicrosoftMQInput** to open its **Component** view.
2. In the **Host** field, type in the host name or address. Here, we type in *localhost*.
3. In the **Queue** field, type in the queue name from which you want to fetch the message. In this example, it is *AddressQueue*.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Press **F6** or click **Run** on the Run tab to execute the Job.
The message body *Atlanta* fetched from the queue is displayed on the console.
tMicrosoftMQOutput

Writes a defined column of given inflow data to Microsoft message queue (only support String type).

**tMicrosoftMQOutput Standard properties**

These properties are used to configure tMicrosoftMQOutput running in the Standard Job framework.
The Standard tMicrosoftMQOutput component belongs to the Internet family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally. Enter properties manually.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file where properties are stored. The fields that come after are pre-filled in using the fetched data.</td>
</tr>
<tr>
<td>Host</td>
<td>Type in the Host name or the IP address of the host server.</td>
</tr>
<tr>
<td>Queue</td>
<td>Type in the name of the queue which you want write a given message in. This queue can be created automatically on the fly if it does not exist then.</td>
</tr>
<tr>
<td>Queue type</td>
<td>Select a Microsoft message queue type from the drop-down list.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Private Non-Transaction</strong>: a private non-transactional queue.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Private Transaction</strong>: a private transactional queue.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Public Non-Transaction</strong>: a public non-transactional queue.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Public Transaction</strong>: a public transactional queue.</td>
</tr>
<tr>
<td>Message column</td>
<td>Select the column as message to be written to Microsoft message queue. The selected column must be of String type.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>
Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component must be linked to an input or intermediary component. |

Connections

Outgoing links (from this component to another):
- **Row**: Main, Iterate
- **Trigger**: Run if, On Component Ok; On Component Error.

Incoming links (from one component to this one):
- **Row**: Main; Reject; Iterate;
- **Trigger**: Run if, On Subjob Ok, On Subjob Error; On Component Ok, On Component Error.

For further information regarding connections, see Talend Studio User Guide.

Limitation

- The message to be output cannot be null.
- This component requires installation of its related jar files.

Related scenario

For a related scenario, see Scenario: Writing and fetching queuing messages from Microsoft message queue on page 2264.
**tMomCommit**

Commits data on the MQ Server.

`tMomCommit` commits in one go, using a unique connection, a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

**tMomCommit Standard properties**

These properties are used to configure `tMomCommit` running in the Standard Job framework.

The Standard `tMomCommit` component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Component List</th>
<th>Select the Connection component used in your Job.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Warning:**

If you want to use a **Row > Main** connection to link `tMomCommit` to your Job, your data will be committed row by row. In this case, do not select the **Close connection** check box or your connection will be closed before the end of your first row commit.

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

For further information about variables, see *Talend Studio User Guide*. |
## Usage

| Usage rule | This component is more commonly used with other tMom* components, especially with the tMomConnection and tMomRollback components. |

## Related scenario

For tMomCommit related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476.
# tMomConnection

Opens a connection to the MQ Server for communication.

## tMomConnection Standard properties

These properties are used to configure tMomConnection running in the Standard Job framework.

The Standard tMomConnection component belongs to the Internet family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Failover</strong></td>
<td>Select this check box to use the ActiveMQ failover transport. For more information about the failover transport, see Failover Transport Reference. This check box is available only when ActiveMQ is selected from the MQ server drop-down list and the Static Discovery check box is cleared.</td>
</tr>
<tr>
<td><strong>Static Discovery</strong></td>
<td>Select this check box to use the ActiveMQ static transport discovery mechanism. For more information about the static transport, see Static Transport Reference. This check box is available only when ActiveMQ is selected from the MQ server list and the Failover check box is cleared.</td>
</tr>
<tr>
<td><strong>URI parameters</strong></td>
<td>Enter the URI parameters. This field is available only when the Failover or Static Discovery check box is selected.</td>
</tr>
<tr>
<td><strong>MQ server</strong></td>
<td>Select the type of the MQ server(s) to be connected from the drop-down list, either ActiveMQ or WebSphere MQ.</td>
</tr>
<tr>
<td><strong>Use SSL Transport</strong></td>
<td>Select this check box to use the ActiveMQ SSL transport. For more information about the SSL transport, see SSL Transport Reference. This check box is available only when ActiveMQ is selected from the MQ server list.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>The host name or IP address of the MQ server.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>The listening port of the MQ server.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>User authentication credentials. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>
### Use Transacted
Select this check box to set the session created to be transacted.
This check box is available only when **ActiveMQ** is selected from the **MQ server** drop-down list.

### Channel
Specify the name of the channel through which the connection is established. The default value is **DC.SVRCONN**.
This field is available only when **WebSphere MQ** is selected from the **MQ server** drop-down list.

### QueueManager
Specify the name of the queue manager to which the connection is established.
This field is available only when **WebSphere MQ** is selected from the **MQ server** drop-down list.

### Use or register a shared Connection
Select this check box to share your connection or use a connection shared by a parent or child Job. This allows you to share one single connection among several connection components from different Job levels that can be either parent or child.
This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the **tRunJob** component. Using a shared connection together with a **tRunJob** component with either of these two options enabled will cause your Job to fail.

### Shared Connection Name
Enter the shared connection name.
This field is available only when the **Use or register a shared Connection** check box is selected.

### Advanced settings
#### Additional options
Define the additional transport options in the table. This table is available only when the **Failover** or **Static Discovery** check box in the **Basic settings** view is selected.

- **Option name**: the name of the option.
- **Option value**: the value of the option.

For example, enter **maxReconnectAttempts** in the **Option name** field and then **0** in the **Option value** field so that no attempt is made to retry the connection when no server is available.

For more information about the failover transport options, see **Failover Transport Reference**.

For more information about the static discovery transport options, see **Static Transport Reference**.

#### tStatCatcher Statistics
Select this check box to collect log data at the component level.
**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it. For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component is more commonly used with other tMom* components, especially with the tMomCommit and tMomRollback components.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component supports ActiveMQ and WebSphere MQ servers.</td>
</tr>
</tbody>
</table>

**Related scenario**

For a similar scenario, see *tMysqlConnection* on page 2475.
**tMomInput**

Fetches a message from a queue on a Message-Oriented Middleware (MOM) system and passes it on to the next component.

*tMomInput* makes it possible to set up asynchronous communications via a MOM server.

**tMomInput Standard properties**

These properties are used to configure *tMomInput* running in the Standard Job framework.

The Standard *tMomInput* component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Use existing connection**      | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.  
Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level. |
| **Keep Listening**               | Select this check box to keep the MOM server listening for and fetching new messages.  
This check box is not visible when **ActiveMQ** is selected from the MQ server list and the receive number of messages check box is selected. |
| **Sleeping time (in sec)**       | Specify the timeout interval (in seconds) within which the next message must be received.  
This field is available in any one of the following situations:  
• JBoss Messaging is selected from the MQ server list and the Keep Listening check box is selected.  
• ActiveMQ is selected from the MQ server list and the Keep Listening check box is cleared.  
• ActiveMQ is selected from the MQ server list and the receive number of messages and Timeout for receiver check boxes are selected. |
| **receive number of messages**   | Select this check box to set the maximum number of messages that can be received.  
This check box is available only when ActiveMQ is selected from the MQ server list and the Keep Listening check box is cleared. |
| **maximum messages**             | Enter the maximum number of messages you can receive.  
This field is available only when the receive number of messages check box is selected. |
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timeout for receiver</strong></td>
<td>Select this check box and specify the timeout interval for the receiver in the <strong>Sleeping time (in sec)</strong> field. This check box is available only when the <strong>receive number of messages</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Failover</strong></td>
<td>Select this check box to use the ActiveMQ failover transport. For more information about the failover transport, see <strong>Failover Transport Reference</strong>. This check box is available only when <strong>ActiveMQ</strong> is selected from the <strong>MQ server</strong> list and the <strong>Static Discovery</strong> check box is cleared.</td>
</tr>
<tr>
<td><strong>Static Discovery</strong></td>
<td>Select this check box to use the ActiveMQ static transport discovery mechanism. For more information about the static transport, see <strong>Static Transport Reference</strong>. This check box is available only when <strong>ActiveMQ</strong> is selected from the <strong>MQ server</strong> list and the <strong>Failover</strong> check box is cleared.</td>
</tr>
<tr>
<td><strong>URI parameters</strong></td>
<td>Enter the URI parameters. This field is available only when the <strong>Failover</strong> or <strong>Static Discovery</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>MQ server</strong></td>
<td>Select the type of the MQ server(s) to be connected from the list: <strong>ActiveMQ</strong>, <strong>JBoss Messaging</strong>, or <strong>WebSphere MQ</strong>. According to the server type selected, the parameters required differ slightly.</td>
</tr>
<tr>
<td><strong>Use SSL Transport</strong></td>
<td>Select this check box to use the ActiveMQ SSL transport. For more information about the SSL transport, see <strong>SSL Transport Reference</strong>. This check box is available only when <strong>ActiveMQ</strong> is selected from the <strong>MQ server</strong> list.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Enter the host name or IP address of the MQ server.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Enter the listening port of the MQ server.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Enter the user authentication credentials. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Channel</strong></td>
<td>Specify the name of the channel through which the connection is established. The default value is <strong>DC.SVRCONN</strong>. This field is available only when <strong>WebSphere MQ</strong> is selected from the <strong>MQ server</strong> list.</td>
</tr>
<tr>
<td><strong>Message From</strong></td>
<td>Type in the message source, exactly as expected by the server; this must include the type and name of the source. e.g.: queue/A or topic/testtopic</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Message Type</strong></td>
<td>Select the message type, either <strong>Topic</strong> or <strong>Queue</strong>. This field is available only when <strong>ActiveMQ</strong> or <strong>JBoss Messaging</strong> is selected from the <strong>MQ Server</strong> list.</td>
</tr>
<tr>
<td><strong>Message Body Type</strong></td>
<td>Select the message body type: <strong>Text Message</strong>, <strong>Bytes Message</strong> or <strong>Map Message</strong>.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. In the context of <strong>tMomInput</strong> usage, the schema is comprised of two columns: <strong>From</strong> and <strong>Message</strong>, and the column names are read only.</td>
</tr>
<tr>
<td><strong>QueueManager</strong></td>
<td>Specify the name of the queue manager to which the connection is established. This field is available only when <strong>WebSphere MQ</strong> is selected from the <strong>MQ server</strong> list.</td>
</tr>
<tr>
<td><strong>MessageQueue</strong></td>
<td>Enter the name of the message queue into which message queueing applications can put messages. This field is available only when <strong>WebSphere MQ</strong> is selected from the <strong>MQ server</strong> list.</td>
</tr>
<tr>
<td><strong>Is using message id to fetch</strong></td>
<td>Select this check box to fetch messages according to their IDs. This check box is available only when <strong>Text Message</strong> or <strong>Bytes Message</strong> is selected from the <strong>Message Body Type</strong> list.</td>
</tr>
<tr>
<td><strong>Commit (delete message after read from the queue)</strong></td>
<td>Select this check box to force a commit after reading each message from the queue. This check box is available only when <strong>WebSphere MQ</strong> is selected from the <strong>MQ server</strong> list and the <strong>Browse message</strong> check box in the <strong>Advanced settings</strong> view is cleared.</td>
</tr>
<tr>
<td><strong>Backout messages to input queue</strong></td>
<td>Select this check box to put the message back into the queue after reading it. This check box and the <strong>Browse message</strong> check box in the <strong>Advanced settings</strong> view enable you to read messages non-destructively from the queue. For further information, see <a href="https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csquizw07.pdf">https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csquizw07.pdf</a>.</td>
</tr>
</tbody>
</table>

Note that the field is case-sensitive.

This field is available only when **ActiveMQ** or **JBoss Messaging** is selected from the **MQ Server** list.
<table>
<thead>
<tr>
<th>Backout messages to backout queue (when backoutCount of message reach the threshold value)</th>
<th>Select this check box to backout the messages to the backout queue defined in the WebSphere MQ manager when the backout count reaches the configured threshold which is a property of the queue in the WebSphere MQ server. This check box is available only when WebSphere MQ is selected from the MQ server list and the Keep Listening, Commit (delete message after read from the queue) and Browse message check boxes are cleared.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Transacted</td>
<td>Select this check box to transact the session. For further information about this parameter, see <a href="https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqzaw07.pdf">https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqzaw07.pdf</a>. <strong>Note:</strong> Selecting this check box will ignore the settings in the Acknowledgement Mode list in the Advanced settings view of tMomInput. This check box is available only when ActiveMQ is selected from the MQ server list.</td>
</tr>
<tr>
<td>start server</td>
<td>Select this check box to force a start of the ActiveMQ server. This check box is available only when ActiveMQ is selected from the MQ server list.</td>
</tr>
</tbody>
</table>

### Advanced settings

| Additional options | Define the additional transport options in the table. This table is available only if you select the Failover or Static Discovery check box in the Basic settings view.  
- Option name: the name of the option.  
- Option value: the value of the option.  
For example, enter maxReconnectAttempts in the Option name field and then 0 in the Option value field so that no attempt is made to retry the connection when no server is available for the failover mode.  
For more information about the failover transport options, see Failover Transport Reference.  
For more information about the static discovery transport options, see Static Transport Reference. |
| Acknowledgement Mode | Select an acknowledgement mode from the list to indicate that the client will acknowledge any messages it receives:  
**Auto Acknowledge:** With this acknowledgement mode, the client automatically acknowledges a message when it has either successfully returned from a call to receive, or the message listener it has called to process the message successfully returns. |
**Client Acknowledge**: With this acknowledgement mode, the client acknowledges a message by calling a message’s acknowledge method.

**Dups OK Acknowledge**: This acknowledgement mode instructs the session to lazily acknowledge the delivery of messages.

For further information about the usage of Jms headers, see [https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqqaw07.pdf](https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqqaw07.pdf).

**Note:**
- If the **Set Transacted** check box of **tMomOutput** is selected, **Acknowledgement Mode** will be ignored. This check box is enabled when the MQ server is **ActiveMQ** or **JBoss Messaging**.

<table>
<thead>
<tr>
<th>Get Jms Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to receive the Jms headers through the mapping from Jms fields onto MQ Series fields. When this check box is checked, you can specify the Jms header and the corresponding reference column name in the line(s) you added by clicking the plus button in the Parameters table. For further information about the usage of Jms headers, see <a href="https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqqaw07.pdf">https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqqaw07.pdf</a>.</td>
</tr>
</tbody>
</table>

**Note:**
- This check box is enabled when the MQ server is **ActiveMQ** or **JBoss Messaging**.

<table>
<thead>
<tr>
<th>Get Jms Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to receive the Jms properties mapped to MQMD fields. When this check box is checked, you can specify the property name, the property type and the reference column name in the line(s) you added by clicking the plus button in the Parameters table. For further information about the usage of Jms properties, see <a href="https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqqaw07.pdf">https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqqaw07.pdf</a>.</td>
</tr>
</tbody>
</table>

(Available when **ActiveMQ** or **JBoss Messaging** is selected in the **MQ Server** list.)

<table>
<thead>
<tr>
<th>Browse message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to disable Commit (delete message after read from the queue) check box and <strong>Backout messages to input queue</strong> check box in the Basic settings view and open the queue to browse messages.</td>
</tr>
</tbody>
</table>

**Note:**
- This check box and the **Backout messages to input queue** check box in the Basic settings view enable you to read messages non-destructively from the queue. **Browse message** check box is visible only when the MQ server is WebSphere MQ with the **Backout messages to input queue** check box cleared. For further information, see [http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0m0/index.jsp?topic=%2FCom.ibm.mq.java.doc%2Fcom%2Fibm%2Fmq%2FmqMD.html](http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0m0/index.jsp?topic=%2FCom.ibm.mq.java.doc%2Fcom%2Fibm%2Fmq%2FmqMD.html).
| Get MQMD Fields | Select this check box to set one or more message descriptors by adding new fields for MQMD (message queuing message descriptor) in the Parameters table:<br><br><strong>Field Name:</strong> Select one or more message descriptors from the list to retrieve header information of the message.<br><br><strong>Reference Column Name:</strong> The header and properties information of the message.<br><br>For further information, see [http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0m0/index.jsp?topic=%2Fcom.ibm.mq.csqzak.doc%2Ffr13040_.htm/](http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0m0/index.jsp?topic=%2Fcom.ibm.mq.csqzak.doc%2Ffr13040_.htm/). (Available when WebSphere MQ is selected in the MQ Server list.) |
|---------------------------------------------------------------|
| Include Header | Select this check box to specify the header of the message:<br><br><strong>MQRFH2 fixed Portion:</strong> Select this check box and click the plus button to add one or more lines to specify the Version, Encoding, Format, Flags, NameValueCCSID and CodedCharSetId as needed for the fixed portion of MQRFH2 header.<br><br><strong>MCD folder:</strong> Select this check box and click the plus button to add one or more lines to specify the fields and the reference column names for the properties that describe the format of the message.<br><br><strong>JMS folder:</strong> Select this check box and click the plus button to add one or more lines to specify the fields and the reference column names for the transportation of JMS header fields and JMSX properties.<br><br><strong>USR folder:</strong> Select this check box and click the plus button to add one or more lines to specify the fields and the reference column names for the transportation of application-defined properties associated with the message.<br><br>For further information about MQRFH2 header, see [http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r1m0/index.jsp?topic=%2Fcom.ibm.etools.mft.doc%2Faq06920_.htm](http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r1m0/index.jsp?topic=%2Fcom.ibm.etools.mft.doc%2Faq06920_.htm). (Available when WebSphere MQ is selected in the MQ Server list.) |
|---------------------------------------------------------------|
| Set CipherSpec | Select this check box to enable the CipherSpec list from which you can specify the CipherSpec to be used with WebSphere MQ SSL.<br><br>For further information about CipherSpec, see [http://publib.boulder.ibm.com/infocenter/wmipv6/v6r0/index.jsp?topic=%2Fcom.ibm.mq.csqzas.doc%2Fsy12870_.htm](http://publib.boulder.ibm.com/infocenter/wmipv6/v6r0/index.jsp?topic=%2Fcom.ibm.mq.csqzas.doc%2Fsy12870_.htm). (Available when WebSphere MQ is selected in the MQ Server list.) |
tStatCatcher Statistics

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

Global Variables

Global Variables

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

This component is generally used as a start component. It must be linked to an output component.

Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your MQ Server connection dynamically from multiple connections planned in your Job.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Limitation

Make sure the relevant ActiveMQ, JBoss Messaging or Websphere MQ server is launched.

This component requires installation of its related jar files.
Scenario 1: Asynchronous communication via a MOM server

This scenario is made of two Jobs. The first Job posts messages on a JBoss server queue and the second Job fetches the message from the server.

Configuring and executing the first Job

About this task

In the first Job, a string message is created using a tRowGenerator and put on a JBoss server using a tMomOutput. An intermediary tLogRow component displays the flow being passed.

Procedure

1. Drop the three components required for the first Job from the Palette onto the design workspace and right-click to connect them using a Main row link.

   ![Diagram of the components](image)

   - tRowGenerator_1 -> row1 (Main) -> tLogRow_1 -> row2 (Main) -> tMomOutput_1

2. Double-click on tRowGenerator to set the schema to be randomly generated.

   ![Schema configuration](image)

3. Set just one column called message. This is the message to be put on the MOM queue.

4. This column is of String type and is nullable. To produce the data, use a preset function which concatenates randomly chosen ascii characters to form a 6-char string. This function is `getAsciiRandomString` (Java version). Click the Preview button to view a random sample of data generated.

5. Set the Number of rows to be generated to 10.

   Click OK to validate.

6. The tLogRow is only used to display an intermediary state of the data to be handled. In this example, it doesn’t require any specific configuration.

7. Then select the tMomOutput component.

   ![MQ server configuration](image)

   MQ server: JBoss Messaging
   Host: localhost
   Port: 1099
   Queue: queueA
   Schema: Built-in

8. In this case, the MQ server to be used is JBoss.
In the **Host** and **Port** fields, fill in the relevant connection information.

9. Select the **Message type** from the list. The message can be of **Queue** or **Topic** type. In this example, select the **Queue** type from the list.

10. In the **To** field, type in the message source information strictly respecting the syntax expected by the server. This should match the Message Type you selected, such as: `queue/A`.

   **Note:**
   The message name is case-sensitive, therefore `queue/A` and `Queue/A` are different.

11. Then click **Sync Columns** to pass on the schema from the preceding component. The schema being read-only, it cannot be changed. The data posted onto the MQ comes from the first schema column encountered.

12. Press **F6** to execute the Job and view the data flow being passed on in the console, thanks to the **tLogRow** component.

---

**Starting job MomInput at 17:46 14/09/2007.**

c8ISGC
LeHc4I
ospGyqP
x6qGt0
ESknhZp
tXU2ET
6H7Nw1
8U9eW
2sebwy
zFZAP

**Job MomInput ended at 17:46 14/09/2007. [exit code=0]**
---

### Configuring and executing the second Job

**About this task**

Then set the second Job in order to fetch the queuing messages from the MOM server.

**Procedure**

1. Drop the **tMomInput** component and a **tLogRow** from the **Palette** to the design workspace.

   ![Diagram](image)

2. Select the **tMomInput** to set the parameters.

   ![Screenshot of settings]

   - **Keep Listening**
   - **Sleeping Time (in sec)**: 5
   - **MQ server**
   - **Host**: 10.42.10.96
   - **Port**: 1099
   - **Message From**: `queue/A`
   - **Message Type**
   - **Queue**
   - **Schema**: **Built-In**

3. Select the **MQ server** from the list. In this example, a JBoss messaging server is used.

4. Set the server **Host** and **Port** information.

5. Set the **Message From** and the **Message Type** to match the source and type expected by the messaging server.
6. The **Schema** is read-only and is made of two columns: **From** and **Message**.

7. Select the **Keep listening** check box and set the verification frequency to 5 seconds.

   **Note:**
   When using the **Keep Listening** option, you'll need to kill the Job to end it.

8. No need to change any default setting from the **tLogRow**.

9. Save the Job and run it (when launching for the first time or if you killed it on a previous run).

   ```
   Starting job momoutput at 17:47 14/05/2007
   [statistics] connecting to socket on port 3414
   [statistics] connected
   Ready to receive message
   Waiting...
   queue/A1c0X5GC
   queue/A11EhC41
   queue/A1opGYqP
   queue/A1x6g0t8
   queue/A1E5kn2p
   queue/A1tXU2ET
   queue/A15k7Vw1
   queue/A18J49eM
   queue/A12s5b9V
   queue/A1rfZP8P
   Job momoutput ended at 17:47 14/05/2007. [exit code=0]
   ```

The messages fetched on the server are displayed on the console.

**Scenario 2: Transmitting XML files via a MOM server**

This scenario describes a five-component Job composed of two subjobs that sends XML files from a local folder to a MOM queue, and then fetches the files from the MOM queue and displays the contents of the files on the console.
Dropping and links the components

Procedure

1. From the Palette, drop the following components one after another onto the design workspace: tFileList, tFileInputXML, tMomOutput, tMomInput, and tLogRow.
2. Connect tFileList to tFileInputXML using a Row > Iterate link, and connect tFileInputXML to tMomOutput using a Row > Main link to form the first subjob. This subjob will read each XML file in a given folder and send it to a MOM queue.
3. Connect tMomInput to tLogRow using a Row > Main link to form the second subjob. This subjob will fetch the XML files from MOM queue and display the file contents on the console.
4. Connect tFileInputXML to tMomInput using a Trigger > On Component Ok connection to link the two subjobs.

Configuring the first subjob

Configuring the input components

Procedure

1. Double-click the tFileList component to open its Basic settings view.

2. In the Directory field, enter the path to the directory to read XML files from, or browse to the path by clicking the [...] button next to the field.
3. Select Use Glob Expressions as Filemask check box, add a new line in the Files field by clicking the [+ ] button, and enter ".*\.xml" as the file mask so that all XML files in the directory will be used. Keep all the other settings as they are.
4. Double-click the tFileInputXML component to open its Basic settings view.
5. Click the [...] button next to Edit schema to open the [Schema] dialog box.

6. Click the [+ ] button to add a column, give it a name, message in this example, and select Document from the Type list to handle XML format files. Then, click OK to close the dialog box.

7. In the File name/Stream field, press Ctrl+Space to access the global variable list, and select tFileList_1.CURRENT_FILEPATH to loop on the context files' directory.

8. In the Loop XPath query fields, enter "/" to define the root as the loop node of the input files' structure; in the Mapping table, fill the XPath query column with ". " to extract all data from context node of the source files, and select the Get Nodes check box to build a Document type data flow.

Configuring the tMomOutput component

Procedure

1. Double-click the tMomOutput component to open its Basic settings view.
2. Select **WebSphere MQ** from the **MQ server** list, and enter the host name or IP address of the MQ server and the port number.

3. Enter the login authentication information in the **Username** and **Password** fields, and enter the channel name of the transmission queue in the **Channel** field.

4. As we are handling file messages, select **Text Message** from the **Message Body Type** list.

5. Click **Sync columns** to retrieve the schema structure from the preceding component.

6. Fill in the queue manager and message queue details in the corresponding fields, and leave the other settings as they are.

### Configuring the second subjob

**Procedure**

1. Double-click the **tMomInput** component to open its **Basic settings** view.

2. Set the basic parameters of the component using the same settings you have done in the **tMomOutput** component, including the MQ server details, login authentication details, channel, message body type, queue manager and message queue.

3. Click the [...] button next to **Edit schema** to open the [Schema] dialog box.
4. From the Type list for the message column, select Document to handle XML format files, and then click OK to close the dialog box.

**Saving and executing the Job**

**Procedure**

1. Press Ctrl+S to save your Job.
2. Press F6 or click Run on the Run tab to execute the Job.

   The XML files in the specified folder are written to the message queue and then retrieved from the queue. The contents of the files are displayed on the console.
<bookstore id="1">
  <bookshelf>
    <category>languages</category>
    <quantity>200</quantity>
  </bookshelf>
  <bookshelf>
    <category>arts</category>
    <quantity>300</quantity>
  </bookshelf>
  <bookshelf>
    <category>science</category>
    <quantity>400</quantity>
  </bookshelf>
</bookstore>

Closing connection
Closing connection
Ready to receive message
Waiting
queue1
<bookstore id="2">
  <bookshelf>
    <category>novels</category>
    <quantity>500</quantity>
  </bookshelf>
  <bookshelf>
    <category>pictorials</category>
    <quantity>100</quantity>
  </bookshelf>
  <bookshelf>
    <category>music</category>
    <quantity>400</quantity>
  </bookshelf>
</bookstore>

Closing connection
**tMomMessageIdList**

Fetches a message ID list from a queue on a Message-Oriented middleware system and passes it to the next component.

`tMomMessageIdList` makes it possible to iterate on certain message IDs. It is usually used with `tMomInput`, for more information, see `tMomInput Standard properties` on page 2276.

**tMomMessageIdList Standard properties**

These properties are used to configure `tMomMessageIdList` running in the Standard Job framework.

The Standard `tMomMessageIdList` component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Use existing connection</strong></th>
<th>Select this check box and from the list displayed select the relevant connection component to reuse the connection details you have already defined. When a Job contains a parent Job and a child Job, the list displayed presents only the connection components in the same Job level.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MQ Server</strong></td>
<td>Select the MOM server to be used from the list. According to the server selected, the parameters required differ slightly.</td>
</tr>
<tr>
<td><strong>Host/Port</strong></td>
<td>Fill in the Host name or IP address of the MOM server and Port.</td>
</tr>
<tr>
<td><strong>Channel</strong></td>
<td>Channel on the queue.</td>
</tr>
<tr>
<td><strong>Queue Manager</strong></td>
<td>Fill in the server driver details.</td>
</tr>
<tr>
<td><strong>Message Queue</strong></td>
<td>Source of the message.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>tStatCatcher Statistics</strong></th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>CURRENT_MESSAGE_ID</strong>: the identifier of the current message. This is a Flow variable and it returns a string.</td>
</tr>
</tbody>
</table>
**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is generally used as an input component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Make sure the relevant Websphere server is launched.</td>
</tr>
<tr>
<td></td>
<td>This component requires installation of its related jar files.</td>
</tr>
</tbody>
</table>

**Related scenario**

For a related scenario, see `tMomInput` on page 2276.
tMomOutput

Adds a message to a Message-Oriented Middleware system queue in order for it to be fetched asynchronously.

tMomOutput makes it possible to set up asynchronous communications via a MOM server.

**tMomOutput Standard properties**

These properties are used to configure tMomOutput running in the Standard Job framework. The Standard tMomOutput component belongs to the Internet family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.</td>
</tr>
<tr>
<td>Failover</td>
<td>Select this check box to use the ActiveMQ failover transport. For more information about the failover transport, see Failover Transport Reference. This check box is available only when ActiveMQ is selected from the MQ server list and the Static Discovery check box is cleared.</td>
</tr>
<tr>
<td>Static Discovery</td>
<td>Select this check box to use the ActiveMQ static transport discovery mechanism. For more information about the static transport, see Static Transport Reference. This check box is available only when ActiveMQ is selected from the MQ server list and the Failover check box is cleared.</td>
</tr>
<tr>
<td>URI parameters</td>
<td>Enter the URI parameters. This field is available only when the Failover or Static Discovery check box is selected.</td>
</tr>
<tr>
<td>MQ Server</td>
<td>Select the type of the MQ server(s) to be connected from the list: ActiveMQ, JBoss Messaging, or WebSphere MQ. According to the server type selected, the parameters required differ slightly.</td>
</tr>
<tr>
<td>Use SSL Transport</td>
<td>Select this check box to use the ActiveMQ SSL transport. For more information about the SSL transport, see SSL Transport Reference. This check box is available only when ActiveMQ is selected from the MQ server list.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Enter the host name or IP address of the MQ server.</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Enter the listening port of the MQ server.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Enter the user authentication credentials. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Channel</strong></td>
<td>Specify the name of the channel through which the connection is established. The default value is <em>DC.SVRCONN</em>. This field is available only when <strong>WebSphere MQ</strong> is selected from the <strong>MQ server</strong> list.</td>
</tr>
<tr>
<td><strong>To</strong></td>
<td>Type in the message destination, respecting the syntax required by the server; this must include the type and name of the target folder, e.g.: queue/A or topic/testtopic. Note that this field is case-sensitive. This field is available only when <strong>ActiveMQ</strong> or <strong>JBoss Messaging</strong> is selected from the <strong>MQ Server</strong> list.</td>
</tr>
<tr>
<td><strong>Message Type</strong></td>
<td>Select the message type: either <strong>Topic</strong> or <strong>Queue</strong>. This list is available only when <strong>ActiveMQ</strong> or <strong>JBoss Messaging</strong> is selected from the <strong>MQ Server</strong> list.</td>
</tr>
<tr>
<td><strong>Message Body Type</strong></td>
<td>Select the message body type: <strong>Text Message</strong>, <strong>Bytes Message</strong> or <strong>Map Message</strong>.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. In the context of <strong>tMomOutput</strong> usage, the schema is read-only but will change according to the incoming schema. Only one-column schema is expected by the server to contain the <strong>Messages</strong>.</td>
</tr>
<tr>
<td><strong>QueueManager</strong></td>
<td>Specify the name of the queue manager to which the connection is established. This field is available only when <strong>WebSphere MQ</strong> is selected from the <strong>MQ server</strong> list.</td>
</tr>
<tr>
<td><strong>MessageQueue</strong></td>
<td>Enter the name of the message queue into which message queueing applications can put messages. This field is available only when <strong>WebSphere MQ</strong> is selected from the <strong>MQ server</strong> list.</td>
</tr>
<tr>
<td><strong>Set Transacted</strong></td>
<td>Select this check box to transact the session. For further information about this parameter, see <a href="https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqzaw07.pdf">https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqzaw07.pdf</a>.</td>
</tr>
</tbody>
</table>
Note that selecting this check box will ignore the settings in the **Acknowledgement Mode** list in the **Advanced settings** view of tMomInput.

| **Is using message id to set** | Select this check box to set messages according to their ids.  
This check box is available only when **Text Message** or **Bytes Message** is selected from the **Message Body Type** list. |
|-------------------------------|----------------------------------------------------------------------------------------------------------|

## Advanced settings

### Additional options

Define the additional transport options in the table. This table is available only if you select the **Failover** or **Static Discovery** check box in the **Basic settings** view.

- **Option name**: the name of the option.
- **Option value**: the value of the option.

For example, enter `maxReconnectAttempts` in the **Option name** field and then `0` in the **Option value** field so that no attempt is made to retry the connection when no server is available.

For more information about the failover transport options, see **Failover Transport Reference**.

For more information about the static discovery transport options, see **Static Transport Reference**.

### Delivery Mode

Select a delivery mode supported by JMS:

- **Not Persistent**: This delivery mode does not require that the message be logged to stable storage.
- **Persistent**: This delivery mode requires that the message be logged to stable storage as part of the client's send operation.


(Available when **ActiveMQ** or **JBoss Messaging** is selected in the **MQ Server** list.)

### Set Jms Header

Select this check box to send the Jms headers through the mapping from Jms fields onto MQ Series fields on the MQ server. When this check box is checked, you can specify the header name and the header value in the line(s) you added by clicking the plus button in the **Parameters** table. For further information about the usage of Jms headers, see [https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqzaw07.pdf](https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqzaw07.pdf).

(Available when **ActiveMQ** or **JBoss Messaging** is selected in the **MQ Server** list.)

### Set Jms Properties

Select this check box to send the Jms properties mapped onto MQMD fields on the MQ server. When this check box is checked, you can specify the property name, the property type and the property value in the line(s) you
added by clicking the plus button in the **Parameters** table. For further information about the usage of Jms properties, see [https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqzaw07.pdf](https://publib.boulder.ibm.com/iseries/v5r2/ic2924/books/csqzaw07.pdf).

(Available when **ActiveMQ** or **JBoss Messaging** is selected in the **MQ Server** list.)

### Use format

Select this check box to specify the WebSphere message format in the **WebSphere Message Format** field. The default format is **MQSTR**.


(Available when **WebSphere MQ** is selected in the **MQ Server** list.)

### Set MQMD Fields

Select this check box to enable the **Parameters** table in which you can specify the value of the MQMD fields.

This check box is available only when **WebSphere MQ** is selected from the **MQ Server** list.

### Parameters

Specify the value of the MQMD fields.

- **Field Name**: Select an MQMD field from the list.
- **Field Value**: Enter the value of the corresponding MQMD field.

This table is available only when the **Set MQMD Fields** check box is selected.

For more information about the MQMD fields, see [Fields for MQMD](http://publib.boulder.ibm.com/infocenter/wmhelp/).

### Include Header

Select this check box to specify the header of the message:

- **MQRFH2 fixed Portion**: Select this check box and click the plus button to add one or more lines to specify the **Version**, **Encoding**, **Format**, **Flags**, **NameValueCCSID** and **CodedCharSetId** as needed for the fixed portion of MQRFH2 header.

- **MCD folder**: Select this check box and click the plus button to add one or more lines to specify the field name and the value for the properties that describe the format of the message.

- **JMS folder**: Select this check box and click the plus button to add one or more lines to specify the field name and the value for the transportation of JMS header fields and JMSX properties.

- **USR folder**: Select this check box and click the plus button to add one or more lines to specify the field name and the value for the transportation of application-defined properties associated with the message.

tMomOutput

<table>
<thead>
<tr>
<th>v6r1m0/index.jsp?topic=%2Fcom.ibm.etools.mft.doc%2Faq06920_.htm. (Available when WebSphere MQ is selected in the MQ Server list.)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set CipherSpec</strong></td>
</tr>
<tr>
<td>Select this check box to enable the CipherSpec list from which you can specify the CipherSpec to be used with WebSphere MQ SSL. For further information about CipherSpec, see <a href="http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=%2Fcom.ibm.mq.csqzas.doc%2Fsy12870_.htm.">http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=%2Fcom.ibm.mq.csqzas.doc%2Fsy12870_.htm.</a> (Available when WebSphere MQ is selected in the MQ Server list.)</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
</tr>
<tr>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

**Global Variables**

<table>
<thead>
<tr>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component must be linked to an input or intermediary component.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dynamic settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your MQ Server connection dynamically from multiple connections planned in your Job.</td>
</tr>
<tr>
<td>The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.</td>
</tr>
</tbody>
</table>
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

**Limitation**

Make sure the relevant Websphere MQ, JBoss Messaging or ActiveMQ server is launched.

This component requires installation of its related jar files.

**Related scenario**

For a related scenario, see tMomInput on page 2276.
# tMomRollback

Cancels the transaction committed in the MQ Server.

## tMomRollback Standard properties

These properties are used to configure tMomRollback running in the Standard Job framework.

The Standard tMomRollback component belongs to the Internet family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Component List</th>
<th>Select the component used in your Job.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Backout messages to backout queue (WebsphereMQ)</strong></td>
<td>Select this check box to backout the messages to the backout queue defined in the Websphere MQ manager when the backout count reaches the configured threshold which is a property of the queue in the Websphere MQ server.</td>
</tr>
<tr>
<td></td>
<td>• If a <strong>tMomConnection</strong> component is selected from Component List, the messages processed by all <strong>tMomInput</strong> components that reuses the connection created by the selected <strong>tMomConnection</strong> component will be backouted.</td>
</tr>
<tr>
<td></td>
<td>• If a <strong>tMomInput</strong> component is selected from Component List and the <strong>Use existing connection</strong> check box in the Basic settings view of the <strong>tMomInput</strong> component is cleared, the messages processed by the selected <strong>tMomInput</strong> component will be backouted.</td>
</tr>
<tr>
<td></td>
<td>• If a <strong>tMomInput</strong> component is selected from Component List and the <strong>Use existing connection</strong> check box in the Basic settings view of the <strong>tMomInput</strong> component is selected, the messages processed by all <strong>tMomInput</strong> components that reuses the same connection the selected <strong>tMomInput</strong> component reused will be backouted.</td>
</tr>
<tr>
<td></td>
<td>• If a <strong>tMomOutput</strong> component is selected from Component List and the <strong>Use existing connection</strong> check box in the Basic settings view of the <strong>tMomOutput</strong> component is cleared, no messages will be backouted.</td>
</tr>
<tr>
<td></td>
<td>• If a <strong>tMomOutput</strong> component is selected from Component List and the <strong>Use existing connection</strong> check box in the Basic settings view of the <strong>tMomOutput</strong> component is selected, the messages processed by all <strong>tMomInput</strong> components that reuses the same connection the selected <strong>tMomOutput</strong> component reused will be backouted.</td>
</tr>
</tbody>
</table>

Note that this check box works only when the **Commit (delete message after read from the queue)**, Backout.
Messages to input queue, Backout messages to backout queue (when backoutCount of message reach the threshold value), and Browse message check boxes of all tMomInput components in the same Job are cleared.

**Close Connection**
Clear this check box to continue to use the selected connection once the component has performed its task.

### Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

### Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

### Usage

| Usage rule | This component is more commonly used with other tMom* components, especially with the tMomConnection and tMomCommit components. |

### Related scenario
For a similar scenario, see Scenario: Rollback from inserting data in mother/daughter tables on page 2480.
tMondrianInput

Executes a multi-dimensional expression (MDX) query corresponding to the dataset structure and schema definition.

tMondrianInput reads data from relational databases and produces multidimensional data sets based on an MDX query. Then it passes on the multidimensional dataset to the next component via a Main row link.

**tMondrianInput Standard properties**

These properties are used to configure tMondrianInput running in the Standard Job framework. The Standard tMondrianInput component belongs to the Business Intelligence family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mondrian Version</td>
<td>Select the Mondrian version you are using.</td>
</tr>
<tr>
<td>DB type</td>
<td>Select the relevant type of relational database</td>
</tr>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the Repository file where Properties are stored. The following fields are pre-filled in using fetched data.</td>
</tr>
<tr>
<td>Datasource</td>
<td>Name and path of the file containing the data.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the</td>
</tr>
</tbody>
</table>
Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see [Talend Studio User Guide].

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see [Talend Studio User Guide].

**Catalog**
Path to the catalog (structure of the data warehouse).

**MDX Query**
Type in the MDX query paying particularly attention to properly sequence the fields in order to match the schema definition and the data warehouse structure.

**Encoding**
Select the encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling.

### Advanced settings

| **tStat Catcher Statistics** | Select this check box to collect log data at the component level. |

### Global Variables

| **Global Variables** | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is a Flow variable and it returns an integer.  
**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see **Talend Studio User Guide**. |

### Usage

| **Usage rule** | This component covers MDX queries for multi-dimensional datasets. |
Limitation
This component requires installation of its related jar files.

Scenario: Cross-join tables

This Job extracts multi-dimensional datasets from relational database tables stored in a MySQL base. The data are retrieved using a multidimensional expression (MDX query). Obviously you need to have to know the structure of your data, or at least have a structure description (catalog) as a reference for the dataset to be retrieved in the various dimensions.

Setting up the Job

Procedure

1. Drop tMondrianInput and tLogRow from the Palette to the design workspace.
2. Connect the Mondrian connector to the output component using a Row Main connection.

Setting up the DB connection

Procedure

1. Double-click the tMondrianInput component to display its Basic settings view.

   - DB Type: MySQL
   - Property Type: Built-In
   - Host: "localhost"
   - Port: 3306
   - Database: "foodmart"
   - User Name: "root"
   - Password: ****
   - Schema: Built-In
   - Catalog: "file:///input/FoodMart.xml"
   - MDX query: "select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]} on columns,
     CrossJoin(
     { [Promotion Media] [All Media] [Radio],
       [Promotion Media] [All Media] [TV],
       [Promotion Media] [All Media] [Sunday Paper],
       [Promotion Media] [All Media] [Street Handout] },
     [Product] [All Products] [Drink].children) on rows
     from Sales
     where ([Time] [1997])"

2. In DB type field, select the relational database you are using with Mondrian.
3. Select the relevant Repository entry as Property type, if you store your DB connection details centrally. In this example the properties are built-in.
4. Fill out the details of connection to your DB: Host, Port, Database name, User Name and Password.
5. Select the relevant Schema in the Repository if you store it centrally. In this example, the schema is to be set (built-in).

### Configuring the DB query

**Procedure**

1. The relational database we want to query contains five columns: media, drink, unit_sales, store_cost and store_sales.
2. The query aims at retrieving the unit_sales, store_cost and store_sales figures for various media / drink using an MDX query such as in the example below:

   **CrossJoin Example 1**

   The current slicer is 1997.

<table>
<thead>
<tr>
<th>Radio</th>
<th>Unit Sales</th>
<th>Store Cost</th>
<th>Store Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alcoholic Beverages</td>
<td>75</td>
<td>70.40</td>
<td>158.62</td>
</tr>
<tr>
<td>Beverages</td>
<td>97</td>
<td>75.70</td>
<td>136.03</td>
</tr>
<tr>
<td>Dairy</td>
<td>54</td>
<td>36.75</td>
<td>39.03</td>
</tr>
<tr>
<td>TV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alcoholic Beverages</td>
<td>76</td>
<td>70.99</td>
<td>132.38</td>
</tr>
<tr>
<td>Beverages</td>
<td>188</td>
<td>167.00</td>
<td>419.14</td>
</tr>
<tr>
<td>Dairy</td>
<td>60</td>
<td>45.19</td>
<td>119.55</td>
</tr>
<tr>
<td>Sunday Paper</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alcoholic Beverages</td>
<td>148</td>
<td>128.97</td>
<td>316.88</td>
</tr>
<tr>
<td>Beverages</td>
<td>197</td>
<td>161.81</td>
<td>399.58</td>
</tr>
<tr>
<td>Dairy</td>
<td>85</td>
<td>54.75</td>
<td>140.27</td>
</tr>
<tr>
<td>Street Handout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alcoholic Beverages</td>
<td>158</td>
<td>121.14</td>
<td>294.55</td>
</tr>
<tr>
<td>Beverages</td>
<td>270</td>
<td>201.28</td>
<td>520.55</td>
</tr>
<tr>
<td>Dairy</td>
<td>84</td>
<td>50.26</td>
<td>128.32</td>
</tr>
</tbody>
</table>

3. Back on the Basic settings tab of the tMondrianInput component, set the Catalog path to the data warehouse. This catalog describes the structure of the warehouse.
4. Then type in the MDX query such as:

   ```mdx
   *select
   ([Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store
```
5. Eventually, select the **Encoding** type on the list.

### Job execution

#### Procedure

1. Select the **tLogRow** component and select the **Print header** check box to display the column names on the console.

2. Then press **F6** to run the Job.

```
Starting job Mondrian at 16:34 08/01/2008.
media drink unit_sales store_cost store_sales
Promotion Media][All Media][Radio][Product][All Products][Drink][Alcoholic beverages][75.0]70.57[100.05
Promotion Media][All Media][Radio][Product][All Products][Drink][Beverages][57.0]75.07[106.03
Promotion Media][All Media][Radio][Product][All Products][Drink][Dairy][54.0]36.74[74.03
Promotion Media][All Media][TV][Product][All Products][Drink][Alcoholic beverages][76.0]50.56[122.38
Promotion Media][All Media][TV][Product][All Products][Drink][Beverages][158.0]157.00[419.14
Promotion Media][All Media][TV][Product][All Products][Drink][Dairy][68.0]45.15[113.55
Promotion Media][All Media][Sunday Paper][Product][All Products][Drink][Alcoholic beverages][149.0]129.57[318.38
Promotion Media][All Media][Sunday Paper][Product][All Products][Drink].
Beverages][197.0]161.88[399.68
Promotion Media][All Media][Sunday Paper][Product][All Products][Drink].
Dairy][85.0]54.24[139.27
Promotion Media][All Media][Street Handout][Product][All Products][Drink][Alcoholic beverages][158.0]121.87[254.35
Promotion Media][All Media][Street Handout][Product][All Products][Drink].
Beverages][270.0]201.28[572.55
Promotion Media][All Media][Street Handout][Product][All Products][Drink].
Dairy][84.0]50.26[134.12
Mondrian ended at 16:51 08/01/2008. Exit code=0.
```

#### Results

The console shows the result of the *unit_sales*, *store_cost* and *store_sales* for each type of *Drink* (*Beverages*, *Dairy*, *Alcoholic beverages*) crossed with each media (*TV*, *Sunday Paper*, *Street handout*) as shown previously in a table form.
tMongoDBBulkLoad

Imports data files in different formats (CSV, TSV or JSON) into the specified MongoDB database so that the data can be further processed.

**tMongoDBBulkLoad Standard properties**

These properties are used to configure tMongoDBBulkLoad running in the Standard Job framework. The Standard tMongoDBBulkLoad component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:
| | • View schema: choose this option to view the schema only.
| | • Change to built-in property: choose this option to change the schema to Built-in for local changes.
| | • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
| MongoDB directory | Fill in this field with the MongoDB home directory. |
| Use local DB path | Select this check box to provide the information of the local database that you want to use. MongoDB V3.0 and onward versions do not support this feature.
| | • Local DB path: type in the path to the local database specified when starting the MongoDB server. |
| Use replica set address | Select this check box to define a replica set to be connected.
| | • Replica set name: specify the name of the replica set.
<p>| | • Replica address: specify multiple MongoDB database servers for failover as needed. Note that if you leave the replica host or replica port unspecified, their default values localhost and 27017 will be used. |</p>
<table>
<thead>
<tr>
<th><strong>Server</strong></th>
<th>Hostname or IP address of the database server. Note that the default value <em>localhost</em> will be used if the server is not specified. This field is available only when the <strong>Use replica set address</strong> check box is not selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port of the database server. Note that the default value 27017 will be used if the port is not specified. This field is available only when the <strong>Use replica set address</strong> check box is not selected.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Type in the name of the database to import data to.</td>
</tr>
<tr>
<td><strong>Collection</strong></td>
<td>Type in the name of the collection to import data to.</td>
</tr>
<tr>
<td><strong>Use SSL connection</strong></td>
<td>Select this check box to enable the SSL or TLS encrypted connection. Then you need to use the <strong>tSetKeystore</strong> component in the same Job to specify the encryption information. For further information about <strong>tSetKeystore</strong>, see <strong>tSetKeystore</strong> on page 3532. Note that the SSL connection is available only for the version 2.4 + of MongoDB.</td>
</tr>
<tr>
<td><strong>Drop collection if exist</strong></td>
<td>Select this check box to remove the collection if it already exists.</td>
</tr>
<tr>
<td><strong>Required authentication</strong></td>
<td>Select this check box to enable the database authentication. Among the mechanisms listed on the <strong>Authentication mechanism</strong> drop-down list, the <strong>NEGOTIATE</strong> one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using. For details about the other mechanisms in this list, see <strong>MongoDB Authentication</strong> from the MongoDB documentation.</td>
</tr>
<tr>
<td><strong>Set Authentication database</strong></td>
<td>If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name of this Authentication database in the <strong>Authentication database</strong> field that is displayed. For further information about the MongoDB Authentication database, see <strong>User Authentication database</strong>.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>
Available when the Required authentication check box is selected.

If the security system you have selected from the Authentication mechanism drop-down list is Kerberos, you need to enter the User principal, the Realm and the KDC server fields instead of the Username and the Password fields.

| Data file | Type in the full path of the file from which the data will be imported or click the [...] button to browse to the desired data file. Make sure that the data file is in standard format. For example, the fields in CSV files should be separated with commas. |
| File type | Select the proper file type from the list. CSV, TSV and JSON are supported. |
| The JSON file starts with an array | Select this check box to allow tMongoDBBulkload to read the JSON files starting with an array. This check box appears when the File type you have selected is JSON. |
| Action on data | Select the action that you want to perform on the data. |
| • Insert: Insert the data into the database. Note that when inserting data from CSV or TSV files into the MongoDB database, you need to specify fields either by selecting the First line is header check box or defining them in the schema. |
| • Upsert: Insert the data if they do not exist or update the existing data. Note that when upserting data into the MongoDB database, you need to specify a list of fields for the query portion of the upsert operation. |
| Upsert fields | Customize the fields that you want to upsert as needed. This table is available when you select Upsert from the Action on data list. |
| First line is header | Select this check box to use the first line in CSV or TSV files as a header. This check box is available only when you select CSV or TSV from the File type list. |
| Ignore blanks | Select this check box to ignore the empty fields in CSV or TSV files. This check box is available only when you select CSV or TSV from the File type list. |
| Print log | Select this check box to print logs. |
Advanced settings

| Additional arguments | Complete this table to use the additional arguments as required. For example, you can use the argument "--jsonArray" to accept the import of data expressed with multiple MongoDB documents within a single JSON array. For more information about the additional arguments, go to [http://docs.mongodb.org/manual/reference/program/mongoimport/](http://docs.mongodb.org/manual/reference/program/mongoimport/) and read the description of options. |

| tStatCatcher Statistics | Select this check box to collect the log data at a component level. |

Global Variables

| Global Variables | NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see [Talend Studio User Guide](http). |

Usage

| Usage rule | This component can be used together with the tMongoDBInput component to check if the data is imported as expected. |

| Limitation | The MongoDB client tool needs to be installed on the machine where Jobs using this component are executed. |

Scenario: Importing data into MongoDB database

This scenario applies only to a Talend solution with Big Data.

The following scenario describes a Job that firstly imports data from a CSV file into the specified MongoDB collection, then reads data from the MongoDB collection to check if the import is successful, next continues to import data from a JSON file with the same data structure into the same MongoDB collection, and finally displays the data from the MongoDB collection to demonstrate that the data from the JSON file is also imported successfully.
Dropping and linking the components

Procedure
1. Drop the following components from the Palette onto the design workspace: two **tMongoDBBulkLoad** components, two **tMongoDBInput** components, and two **tLogRow** components.
2. Connect the first **tMongoDBBulkLoad** to the first **tMongoDBInput** using a **Trigger > OnSubjobOk** link.
3. Connect the first **tMongoDBInput** to the first **tLogRow** using a **Row > Main** link.
4. Repeat the two steps above to connect the second **tMongoDBBulkLoad** to the second **tMongoDBInput**, and the second **tMongoDBInput** to the second **tLogRow**.
5. Connect the first **tMongoDBInput** to the second **tMongoDBBulkLoad** using a **Trigger > OnSubjobOk** link.
6. Label the two **tLogRow** components to better identify the data displayed on the console.

Configuring the components

Importing data from a CSV file

Procedure
1. Double-click the first **tMongoDBBulkLoad** component to open its **Basic settings** view in the **Component** tab.
2. In the MongoDB directory field, type in the MongoDB home directory. In this example, it is D:/MongoDB.

3. In the Server and Port fields, fill in the information required for the connection to MongoDB. In this example, type in localhost and 27017.

4. In the Database field, type in the database to import data to, bookstore in this example.

5. In the Collection field, type in the collection to import data to, books in this example.

6. Select the Drop collection if exist check box to remove the specified collection if it already exists.

7. Browse to the desired data file from which you want to import data. In this example, it is D:/Input/books.csv, which is a standard CSV file containing four columns: id, title, author, and category.

8. Select CSV from the File type list.

9. Select Insert from the Action on data list.

10. Select the First line is header check box to use the first line in the CSV file as a header.

11. Select the Ignore blanks check box to ignore the blank fields (if any) in the CSV file.

Validating that the CSV file is imported successfully

Procedure

1. Double-click the first tMongoDBInput component to open its Basic settings view in the Component tab.
2. From the DB Version list, select the MongoDB version you are using.

3. In the Server and Port fields, fill in the information required for the connection to MongoDB. In this example, type in localhost and 27017.

4. In the Database field, type in the database from which the data will be read, bookstore in this example.

5. In the Collection field, type in the collection from which the data will be read, books in this example.

6. Click Edit schema to define the data structure to be read from the MongoDB collection.
7. In the **Mapping** table, the **Column** field is automatically populated with the defined schema. You do not need to fill in the **Parent node path** column.

8. Double-click the first **tLogRow** component to open its **Basic settings** view in the **Component** tab.

9. In the **Mode** area, select **Table (print values in cells of a table)**.

**Importing data from a JSON file**

**Procedure**

1. Double-click the second **tMongoDBBulkLoad** component to open its **Basic settings** view in the **Component** tab.
2. In the **MongoDB directory** field, type in the MongoDB home directory. In this example, it is `D:/MongoDB`.

3. In the **Server** and **Port** fields, fill in the information required for the connection to MongoDB. In this example, type in `localhost` and `27017`.

4. In the **Database** field, type in the target database to import data, `bookstore` in this example.

5. In the **Collection** field, type in the target collection to import data, `books` in this example.

6. Browse to the desired data file from which you want to import data. Here, select `books.json`.

   ```json
   { "id": "4", "title": "Les Miserables", "author": "Victor Hugo", "category": "Language&Literature" } 
   { "id": "5", "title": "Advanced Database Systems", "author": "Carlo Zaniolo", "category": "Database" }
   ```

7. Select **JSON** from the **File type** list.

8. Select **Insert** from the **Action on data** list.

9. Click the **Advanced settings** tab to define the additional arguments as needed.

   In this example, add the argument `"--jsonArray"` to accept the imported data within a single JSON array.
Validating that the JSON file is imported successfully

Procedure

1. Repeat Step 1 through Step 7 described in the procedure Validating that the CSV file is imported successfully on page 2311 to configure the second tMongoDBInput component.

2. Repeat Step 8 through Step 9 described in the procedure Validating that the CSV file is imported successfully on page 2311 to configure the second tLogRow component.

Saving and executing the Job

Procedure

1. Press Ctrl + S to save the Job.
2. Execute the Job by pressing F6 or clicking Run on the Run tab.
The data from the collection `books` in the MongoDB database `bookstore` is displayed on the console, which contains the data imported from both the CSV file `books.csv` and the JSON file `books.json`. 
tMongoDBClose

Closes a connection to the MongoDB database.

**tMongoDBClose Standard properties**

These properties are used to configure tMongoDBClose running in the Standard Job framework.

The Standard tMongoDBClose component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Component list | Select the tMongoDBConnection component in the list if more than one connection is created for the current Job. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

**Global Variables**

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

**Usage**

**Usage rule**

This component is to be used along with other MongoDB components, especially tMongoDBConnection.

**Related scenario**

For a related scenario, see Scenario 1: Creating a collection and writing data to it on page 2361.
tMongoDBConnection

Creates a connection to a MongoDB database and reuse that connection in other components.
tMongoDBConnection opens a connection to a database in order that a transaction may be made.

**tMongoDBConnection Standard properties**

These properties are used to configure tMongoDBConnection running in the Standard Job framework. 
The Standard tMongoDBConnection component belongs to the Big Data and the Databases families. 
The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB Version</strong></td>
<td>List of the database versions.</td>
</tr>
<tr>
<td><strong>Use replica set address</strong></td>
<td>Select this check box to show the Replica address table. In the Replica address table, you can define multiple MongoDB database servers for failover.</td>
</tr>
<tr>
<td><strong>Server and Port</strong></td>
<td>IP address and listening port of the database server. Available when the Use replica set address check box is not selected.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Use SSL connection</strong></td>
<td>Select this check box to enable the SSL or TLS encrypted connection. Then you need to use the tSetKeystore component in the same Job to specify the encryption information. For further information about tSetKeystore, see tSetKeystore on page 3532. Note that the SSL connection is available only for the version 2.4 + of MongoDB.</td>
</tr>
<tr>
<td><strong>Required authentication</strong></td>
<td>Select this check box to enable the database authentication. Among the mechanisms listed on the Authentication mechanism drop-down list, the NEGOTIATE one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using. For details about the other mechanisms in this list, see MongoDB Authentication from the MongoDB documentation.</td>
</tr>
<tr>
<td><strong>Set Authentication database</strong></td>
<td>If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name</td>
</tr>
</tbody>
</table>
of this Authentication database in the **Authentication database** field that is displayed.

For further information about the MongoDB Authentication database, see **User Authentication database**.

| Username and Password | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. Available when the **Required authentication** check box is selected. If the security system you have selected from the **Authentication mechanism** drop-down list is Kerberos, you need to enter the **User principal**, the **Realm** and the **KDC server** fields instead of the **Username** and the **Password** fields. |

| Advanced settings | Select this check box to collect the log data at a component level. |

| No query timeout | Select this check box to prevent MongoDB servers from stopping idle cursors at the end of 10-minute inactivity of these cursors. In this situation, an idle cursor will stay open until either the results of this cursor are exhausted or you manually close it using the `cursor.close()` method. A cursor for MongoDB is a pointer to the result set of a query. By default, that is to say, with this check box being clear, a MongoDB server automatically stops idle cursors after a given inactivity period to avoid excess memory use. For further information about MongoDB cursors, see [https://docs.mongodb.org/manual/core/cursors/](https://docs.mongodb.org/manual/core/cursors/). |

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see **Talend Studio User Guide**. |

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see **Talend Studio User Guide**. |
Usage

| Usage rule | This component is generally used with other MongoDB components, particularly tMongoClose. |

Related scenario

For a related scenario, see Scenario 1: Creating a collection and writing data to it on page 2361.
tMongoDBGridFSDelete

Automates the delete action over specific files in MongoDB GridFS.

tMongoDBGridFSDelete connects to a given MongoDB GridFS system, executes a user-defined query to select specific files and deletes these files from this system.

**tMongoDBGridFSDelete Standard properties**

These properties are used to configure tMongoDBGridFSDelete running in the Standard Job framework.

The Standard tMongoDBGridFSDelete component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Property type                      | Either **Built-In** or **Repository**.  
|-----------------------------------|------------------------------------------|
| **Built-In**: No property data stored centrally.  
| **Repository**: Select the repository file where the properties are stored.  |

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

Note that when a Job contains the parent Job and the child Job, **Component List** presents only the connection components in the same Job level.

**Use replica set address or multiple query routers**

Select this check box to show the **Server addresses** table.

In the **Server addresses** table, define the sharded MongoDB databases or the MongoDB replica sets you want to connect to.

**Server and Port**

IP address and listening port of the database server.

Available when the **Use replica set address** check box is not selected.

Note that if you use the authentication mechanisms to connect to this MongoDB database, you must enter the name, rather than the IP address, of the host of the database server.

**Database**

Name of the database.

**Use SSL connection**

Select this check box to enable the SSL or TLS encrypted connection.

Then you need to use the **tSetKeystore** component in the same Job to specify the encryption information.
For further information about **tSetKeystore**, see [tSetKeystore](#) on page 3532.

Note that the SSL connection is available only for the version 2.4 + of MongoDB.

<table>
<thead>
<tr>
<th>Set read preference</th>
<th>Select this check box and from the <strong>Read preference</strong> drop-down list that is displayed, select the member to which you need to direct the read operations. If you leave this check box clear, the Job uses the default Read preference, that is to say, uses the primary member in a replica set. For further information, see MongoDB’s documentation about Replication and its Read preferences.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Use authentication</th>
<th>Select this check box to enable the database authentication. Among the mechanisms listed on the <strong>Authentication mechanism</strong> drop-down list, the <strong>NEGOTIATE</strong> one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using. For details about the other mechanisms in this list, see MongoDB Authentication from the MongoDB documentation.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Set Authentication database</th>
<th>If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name of this Authentication database in the <strong>Authentication database</strong> field that is displayed. For further information about the MongoDB Authentication database, see User Authentication database.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Username and Password</th>
<th>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. Available when the <strong>Required authentication</strong> check box is selected. If the security system you have selected from the <strong>Authentication mechanism</strong> drop-down list is Kerberos, you need to enter the <strong>User principal</strong>, the <strong>Realm</strong> and the <strong>KDC server</strong> fields instead of the <strong>Username</strong> and the <strong>Password</strong> fields.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bucket</th>
<th>Enter the name of the bucket from which you need to delete files. A bucket of GridFS is similar to a folder.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Query type</th>
<th>Select the type of the query you want <strong>tMongoDBGridFSDelete</strong> to perform to select the files to be deleted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>Filename</strong>: you need to enter the name of the file to be deleted. This name is the value of the filename</td>
<td></td>
</tr>
</tbody>
</table>
attribute of this file. Note that only one name is allowed.

In GridFS, a file is identified only by its unique identifier (its object ID); its file name is not necessarily unique.

- **Mongo Query**: enter the query to be used to select the files to be deleted.

  The example query, `{}` within double quotation marks provided with `tMongoDBGridFSDelete`, means to select all of the files in a bucket. You can put `{'filename': {'$regex': 'REGEX_PATTERN'}}` to apply a regular expression to defining the file names to be used.

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect the log data at the component level.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No query timeout</strong></td>
<td>Select this check box to prevent MongoDB servers from stopping idle cursors at the end of 10-minute inactivity of these cursors. In this situation, an idle cursor will stay open until either the results of this cursor are exhausted or you manually close it using the <code>cursor.close()</code> method.</td>
</tr>
<tr>
<td></td>
<td>A cursor for MongoDB is a pointer to the result set of a query. By default, that is to say, with this check box being clear, a MongoDB server automatically stops idle cursors after a given inactivity period to avoid excess memory use. For further information about MongoDB cursors, see <a href="https://docs.mongodb.org/manual/core/cursors/">https://docs.mongodb.org/manual/core/cursors/</a>.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_FILE</strong>: the number of files processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <a href="https://docs.talend.com/studio-user-guide">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>
**Usage**

| Usage rule | This component combines MongoDB GridFS connection and data iteration, thus usually used as a single-component Subjob to delete selected files from a given bucket in MongoDB GridFS. It is often connected to the Job using **OnSubjobOk** or **OnComponentOk** link, depending on the context. |

**Related scenario**

For a scenario using `tMongoDBGridFSDelete`, see *Managing files using MongoDB GridFS* on page 2340.
tMongoDBGridFSGet

Connects to a MongoDB GridFS system to copy files from it.
tMongoDBGridFSGet copies files from a given MongoDB GridFS system to a local directory and if needs be, renames these files.

**tMongoDBGridFSGet Standard properties**

These properties are used to configure tMongoDBGridFSGet running in the Standard Job framework. The Standard tMongoDBGridFSGet component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property type</strong></td>
<td>Either Built-In or Repository. Built-In: No property data stored centrally. Repository: Select the repository file where the properties are stored.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.</td>
</tr>
<tr>
<td><strong>Use replica set address or multiple query routers</strong></td>
<td>Select this check box to show the Server addresses table. In the Server addresses table, define the sharded MongoDB databases or the MongoDB replica sets you want to connect to.</td>
</tr>
<tr>
<td><strong>Server and Port</strong></td>
<td>IP address and listening port of the database server. Available when the Use replica set address check box is not selected. Note that if you use the authentication mechanisms to connect to this MongoDB database, you must enter the name, rather than the IP address, of the host of the database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Use SSL connection</strong></td>
<td>Select this check box to enable the SSL or TLS encrypted connection. Then you need to use the tSetKeystore component in the same Job to specify the encryption information. For further information about tSetKeystore, see tSetKeystore on page 3532.</td>
</tr>
<tr>
<td><strong>Set read preference</strong></td>
<td>Note that the SSL connection is available only for the version 2.4 + of MongoDB.</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Set read preference</strong></td>
<td>Select this check box and from the <strong>Read preference</strong> drop-down list that is displayed, select the member to which you need to direct the read operations.</td>
</tr>
<tr>
<td><strong>Set read preference</strong></td>
<td>If you leave this check box clear, the Job uses the default Read preference, that is to say, uses the primary member in a replica set.</td>
</tr>
<tr>
<td><strong>Set read preference</strong></td>
<td>For further information, see MongoDB's documentation about Replication and its Read preferences.</td>
</tr>
<tr>
<td><strong>Required authentication</strong></td>
<td>Select this check box to enable the database authentication.</td>
</tr>
<tr>
<td><strong>Required authentication</strong></td>
<td>Among the mechanisms listed on the <strong>Authentication mechanism</strong> drop-down list, the <strong>NEGOTIATE</strong> one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using.</td>
</tr>
<tr>
<td><strong>Required authentication</strong></td>
<td>For details about the other mechanisms in this list, see MongoDB Authentication from the MongoDB documentation.</td>
</tr>
<tr>
<td><strong>Set Authentication database</strong></td>
<td>If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name of this Authentication database in the <strong>Authentication database</strong> field that is displayed.</td>
</tr>
<tr>
<td><strong>Set Authentication database</strong></td>
<td>For further information about the MongoDB Authentication database, see User Authentication database.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Available when the <strong>Required authentication</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>If the security system you have selected from the <strong>Authentication mechanism</strong> drop-down list is Kerberos, you need to enter the <strong>User principal</strong>, the <strong>Realm</strong> and the <strong>KDC server</strong> fields instead of the <strong>Username</strong> and the <strong>Password</strong> fields.</td>
</tr>
<tr>
<td><strong>Bucket</strong></td>
<td>Enter the name of the bucket you need to read files from. A bucket of GridFS is similar to a folder.</td>
</tr>
<tr>
<td><strong>Query type</strong></td>
<td>Select the type of the query you want <strong>tMongoDBGridFSGet</strong> to perform to select the data to be copied.</td>
</tr>
<tr>
<td><strong>Query type</strong></td>
<td>• <strong>Filename</strong>: you need to enter the name of the file to be copied. This name is the value of the <strong>filename</strong> attribute used by this file. Note that only one name is allowed.</td>
</tr>
</tbody>
</table>
In GridFS, a file is identified only by its unique identifier (its object ID); its file name is not necessarily unique.

- **Mongo Query**: enter the query to be used to select the files to be copied.

  The default query, `{}` within double quotation marks provided with this component, means to select all of the files. You can also apply a regular expression by putting `{'filename':{'$regex':'REGEX_PATTERN'}` to define the file names to be used.

  It is recommended to use your query along with the **Overwrite local files** and the **Use Document ID as output filename** check boxes to avoid any file name conflict issue.

| **Local Folder** | Enter the path to the folder in which the data copied from GridFS is to be written. |
|**Overwrite local files** | Select this check box to overwrite the file using the same file name in the local folder being used. Otherwise, an exception will be returned in this type of situation. |
|**Use Document ID as output filename** | Select this box to rename the incoming files on the fly using the values of their object IDs. This allows you to keep the files that have the same file names. |

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to collect the log data at the component level. |
|**No query timeout** | Select this check box to prevent MongoDB servers from stopping idle cursors at the end of 10-minute inactivity of these cursors. In this situation, an idle cursor will stay open until either the results of this cursor are exhausted or you manually close it using the cursor.close() method.  

  A cursor for MongoDB is a pointer to the result set of a query. By default, that is to say, with this check box being clear, a MongoDB server automatically stops idle cursors after a given inactivity period to avoid excess memory use. For further information about MongoDB cursors, see [https://docs.mongodb.org/manual/core/cursors/](https://docs.mongodb.org/manual/core/cursors/). |

**Global Variables**

| **Global Variables** | **NB_FILE**: the number of files processed. This is an After variable and it returns an integer.  

  **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. |
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component combines MongoDB GridFS connection and data extraction, thus usually used as a single-component subjob to copy data from GridFS to a user-defined local directory. It is often connected to the Job using <strong>OnSubjobOk</strong> or <strong>OnComponentOk</strong> link, depending on the context.</td>
</tr>
</tbody>
</table>

### Related scenario

For a scenario using tMongoDBGridFSGet, see *Managing files using MongoDB GridFS* on page 2340.
tMongoDBGridFSList

Retrieves a list of files based on a query.

tMongoDBGridFSList iterates on files of a bucket in MongoDB GridFS.

**tMongoDBGridFSList Standard properties**

These properties are used to configure tMongoDBGridFSList running in the Standard Job framework. The Standard tMongoDBGridFSList component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.

**Use replica set address or multiple query routers**

Select this check box to show the Server addresses table.

In the Server addresses table, define the sharded MongoDB databases or the MongoDB replica sets you want to connect to.

**Server and Port**

IP address and listening port of the database server.

Available when the Use replica set address check box is not selected.

Note that if you use the authentication mechanisms to connect to this MongoDB database, you must enter the name, rather than the IP address, of the host of the database server.

**Database**

Name of the database.

**Use SSL connection**

Select this check box to enable the SSL or TLS encrypted connection.

Then you need to use the tSetKeystore component in the same Job to specify the encryption information.

For further information about tSetKeystore, see tSetKeystore on page 3532.
Note that the SSL connection is available only for the version 2.4 + of MongoDB.

| Set read preference          | Select this check box and from the **Read preference** drop-down list that is displayed, select the member to which you need to direct the read operations.  
|                             | If you leave this check box clear, the Job uses the default Read preference, that is to say, uses the primary member in a replica set.  
|                             | For further information, see MongoDB's documentation about Replication and its Read preferences. |

| Use authentication          | Select this check box to enable the database authentication.  
|                            | Among the mechanisms listed on the **Authentication mechanism** drop-down list, the NEGOTIATE one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using.  
|                            | For details about the other mechanisms in this list, see MongoDB Authentication from the MongoDB documentation. |

| Set Authentication database | If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name of this Authentication database in the **Authentication database** field that is displayed.  
|                            | For further information about the MongoDB Authentication database, see User Authentication database. |

| Username and Password       | DB user authentication data.  
|                            | To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.  
|                            | Available when the **Required authentication** check box is selected.  
|                            | If the security system you have selected from the **Authentication mechanism** drop-down list is Kerberos, you need to enter the User principal, the Realm and the KDC server fields instead of the Username and the Password fields. |

| Bucket                      | Enter the name of the bucket in which the files to be listed are stored. A bucket of GridFS is similar to a folder. |

| Query                       | Enter the query to be used to select the files to be listed. You need to define how these files are sorted in the list by selecting:  
|                            | • **Sort by**: these files can be sorted by their filename attributes, uploadDate attributes or **length** (the File Size option) attributes. |
• **Sort order**: these files can be sorted in the **Ascending** or **Descending** order.

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect the log data at the component level.</th>
</tr>
</thead>
</table>

**No query timeout**

Select this check box to prevent MongoDB servers from stopping idle cursors at the end of 10-minute inactivity of these cursors. In this situation, an idle cursor will stay open until either the results of this cursor are exhausted or you manually close it using the `cursor.close()` method.

A cursor for MongoDB is a pointer to the result set of a query. By default, that is to say, with this check box being clear, a MongoDB server automatically stops idle cursors after a given inactivity period to avoid excess memory use. For further information about MongoDB cursors, see [https://docs.mongodb.org/manual/core/cursors/](https://docs.mongodb.org/manual/core/cursors/).

### Global Variables

| Global Variables | **NB_FILE**: the number of files processed. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
**CURRENT_FILEID**: the value of the `ObjectId` (the file ID) attribute of the current file. This is an Flow variable and it returns a string.  
**CURRENT_FILENAME**: the value of the `filename` attribute of the current file. This is an Flow variable and it returns a string.  
**CURRENT_FILEUPLOADDATE**: the value of the `uploadDate` attribute of the current file. This is an Flow variable and it returns a date.  
**CURRENT_FILELENGTH**: the value of the `length` (the size) attribute of the current file. This is an Flow variable and it returns a length.  
**CURRENT_FILEMD5**: the value of the `md5` attribute of the current file. This is an Flow variable and it returns a string.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. |
For further information about variables, see *Talend Studio User Guide*.

## Usage

### Usage rule

This component combines MongoDB GridFS connection and data iteration, thus usually used as a single-component Subjob to list files stored in a given bucket in MongoDB GridFS.

Different lists are returned, each representing an attribute of these files and being stored in a specific global variable as explained above regarding the global variables provided by *tMongoDBGridFSList*.

It is often connected to the Job using *OnSubjobOk* or *OnComponentOk* link, depending on the context.

## Related scenario

For a scenario using *tMongoDBGridFSList*, see *Managing files using MongoDB GridFS* on page 2340.
tMongoDBGridFSProperties

Obtains information about the properties of given files selected based on a query. tMongoDBGridFSProperties iterates on files of a bucket in MongoDB GridFS to extract the attributes of these files.

**tMongoDBGridFSProperties Standard properties**

These properties are used to configure tMongoDBGridFSProperties running in the Standard Job framework.

The Standard tMongoDBGridFSProperties component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Property type**                                  | Either **Built-In** or **Repository**.  
**Built-In**: No property data stored centrally.  
**Repository**: Select the repository file where the properties are stored. |
| **Use an existing connection**                    | Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.  
Note that when a Job contains the parent Job and the child Job, **Component List** presents only the connection components in the same Job level. |
| **Use replica set address or multiple query routers** | Select this check box to show the **Server addresses** table.  
In the **Server addresses** table, define the sharded MongoDB databases or the MongoDB replica sets you want to connect to. |
| **Server and Port**                               | IP address and listening port of the database server.  
Available when the **Use replica set address** check box is not selected.  
Note that if you use the authentication mechanisms to connect to this MongoDB database, you must enter the name, rather than the IP address, of the host of the database server. |
| **Database**                                      | Name of the database.                                                                                                                                                                                        |
| **Use SSL connection**                            | Select this check box to enable the SSL or TLS encrypted connection.  
Then you need to use the **tSetKeystore** component in the same Job to specify the encryption information. |


For further information about **tSetKeystore**, see [tSetKeystore](#) on page 3532.

Note that the SSL connection is available only for the version 2.4 + of MongoDB.

| **Set read preference** | Select this check box and from the **Read preference** drop-down list that is displayed, select the member to which you need to direct the read operations.

If you leave this check box clear, the Job uses the default Read preference, that is to say, uses the primary member in a replica set.

For further information, see MongoDB's documentation about Replication and its Read preferences. |
|-------------------------|--------------------------------------------------------------------------------------------------|

| **Use authentication** | Select this check box to enable the database authentication.

Among the mechanisms listed on the **Authentication mechanism** drop-down list, the **NEGOTIATE** one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using.

For details about the other mechanisms in this list, see [MongoDB Authentication](#) from the MongoDB documentation. |
|-----------------------|----------------------------------------------------------------------------------------------------------------------|

| **Set Authentication database** | If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name of this Authentication database in the **Authentication database** field that is displayed.

For further information about the MongoDB Authentication database, see [User Authentication database](#). |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------|

| **Username and Password** | DB user authentication data.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

Available when the **Required authentication** check box is selected.

If the security system you have selected from the **Authentication mechanism** drop-down list is Kerberos, you need to enter the **User principal**, the **Realm** and the **KDC server** fields instead of the **Username** and the **Password** fields. |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------|

<table>
<thead>
<tr>
<th><strong>Bucket</strong></th>
<th>Enter the name of the bucket in which the files to be used are stored. A bucket of GridFS is similar to a folder.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Schema and Edit schema</strong></th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</th>
</tr>
</thead>
</table>
The columns of the predefined schema of **tMongoDBGridFSProperties** represent the default attributes of a GridFS file and the column names are already identical with the attribute names.

- If you want to exclude some attributes from the data flow to be processed in your Job, you can remove the columns corresponding to those attributes from this schema.
- Do not change the names of the columns to be used, because the names of each column must be identical with the name of the file attribute it represents.
- If the files to be processed contain some custom attributes you want to extract, you need to add their representative columns accordingly and ensure that these columns and these custom attributes are using identical names.

### Query type

Select the type of the query you want **tMongoDBGridFSProperties** to perform to select the files from which you need to extract the attributes.

- **Filename**: you need to enter the name of the file to be used. This name is the value of the *filename* attribute of this file. Note that only one name is allowed.

  In GridFS, a file is identified only by its unique identifier (its object ID); its file name is not necessarily unique.

- **Mongo Query**: enter the query to be used to select the files to be used.

  The example query, `{}` within double quotation marks provided with **tMongoDBGridFSProperties**, means to select all of the files in a bucket. You can put `{filename: {'$regex': 'REGEX_PATTERN'}` to apply a regular expression to defining the file names to be used.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to collect the log data at the component level.

**No query timeout**

Select this check box to prevent MongoDB servers from stopping idle cursors at the end of 10-minute inactivity of these cursors. In this situation, an idle cursor will stay open until either the results of this cursor are exhausted or you manually close it using the `cursor.close()` method.

A cursor for MongoDB is a pointer to the result set of a query. By default, that is to say, with this check box being clear, a MongoDB server automatically stops idle cursors after a given inactivity period to avoid excess memory use. For further information about MongoDB cursors, see [https://docs.mongodb.org/manual/core/cursors/](https://docs.mongodb.org/manual/core/cursors/).
Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_FILE</td>
<td>the number of files processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This component combines MongoDB GridFS connection and data extraction and needs to use a Main link to connect to its following component in order to send data to this component.</td>
</tr>
</tbody>
</table>

Related scenario

For a scenario using tMongoDBGridFSProperties, see Managing files using MongoDB GridFS on page 2340.
tMongoDBGridFSPut

Connects to a MongoDB GridFS system to load files into it.

`tMongoDBGridFSPut` copies files from a local directory, pastes them into a given MongoDB GridFS system and if needs be, renames these files.

**tMongoDBGridFSPut Standard properties**

These properties are used to configure `tMongoDBGridFSPut` running in the Standard Job framework. The Standard `tMongoDBGridFSPut` component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the properties are stored.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

Note that when a Job contains the parent Job and the child Job, **Component List** presents only the connection components in the same Job level.

**Use replica set address or multiple query routers**

Select this check box to show the **Server addresses** table.

In the **Server addresses** table, define the sharded MongoDB databases or the MongoDB replica sets you want to connect to.

**Server and Port**

IP address and listening port of the database server.

Available when the **Use replica set address** check box is not selected.

Note that if you use the authentication mechanisms to connect to this MongoDB database, you must enter the name, rather than the IP address, of the host of the database server.

**Database**

Name of the database.

**Use SSL connection**

Select this check box to enable the SSL or TLS encrypted connection.

Then you need to use the `tSetKeystore` component in the same Job to specify the encryption information.

For further information about `tSetKeystore`, see `tSetKeystore` on page 3532.
### Set read preference
Select this check box and from the **Read preference** drop-down list that is displayed, select the member to which you need to direct the read operations.

If you leave this check box clear, the Job uses the default Read preference, that is to say, uses the primary member in a replica set.

For further information, see MongoDB’s documentation about Replication and its Read preferences.

### Required authentication
Select this check box to enable the database authentication.

Among the mechanisms listed on the **Authentication mechanism** drop-down list, the **NEGOTIATE** one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using.

For details about the other mechanisms in this list, see [MongoDB Authentication](https://docs.mongodb.com/v3.2/configservers/kerberos/) from the MongoDB documentation.

### Set Authentication database
If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name of this Authentication database in the **Authentication database** field that is displayed.

For further information about the MongoDB Authentication database, see [User Authentication database](https://docs.mongodb.com/v3.2/core/security/authentication/#using-mongodb-user-authentication).

### Username and Password
DB user authentication data.

To enter the password, click the `[...]` button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

Available when the **Required authentication** check box is selected.

If the security system you have selected from the Authentication mechanism drop-down list is Kerberos, you need to enter the **User principal**, the **Realm** and the **KDC server** fields instead of the **Username** and the **Password** fields.

### Bucket
Enter the name of the bucket you need to write files in. A bucket of GridFS is similar to a folder.

### Local Folder
Browse to, or enter the path to the folder in which the files to be copied and written to GridFS are stored.

### Use Perl5 Regex Expression as Filemask
Select this check box if you want to use Perl5 regular expressions in the **Files** field as file filters. This is useful when the name of the file to be used contains special characters such as parentheses.

Note that the SSL connection is available only for the version 2.4 + of MongoDB.
**Files**

In the **Files** area, the fields to be completed are:
- **File mask**: type in the file name to be selected from the local directory. Regular expression is available.
- **New name**: give a new name to the loaded file.

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect the log data at the component level.</th>
</tr>
</thead>
</table>
| **No query timeout**    | Select this check box to prevent MongoDB servers from stopping idle cursors at the end of 10-minute inactivity of these cursors. In this situation, an idle cursor will stay open until either the results of this cursor are exhausted or you manually close it using the `cursor.close()` method.  
A cursor for MongoDB is a pointer to the result set of a query. By default, that is to say, with this check box being clear, a MongoDB server automatically stops idle cursors after a given inactivity period to avoid excess memory use. For further information about MongoDB cursors, see [https://docs.mongodb.org/manual/core/cursors/](https://docs.mongodb.org/manual/core/cursors/). |

**Global Variables**

| Global Variables | **NB_FILE**: the number of files processed. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see [Talend Studio User Guide](#). |

**Usage**

| Usage rule | This component combines MongoDB GridFS connection and data extraction, thus usually used as a single-component subjob to copy data from a user-defined local directory to GridFS. |
Managing files using MongoDB GridFS

This scenario applies only to a Talend solution with Big Data.

In this scenario, the MongoDB GridFS components are used to create a Job to manage video files in MongoDB GridFS.

For further information about the GridFS system of MongoDB, see When to use GridFS.

For demonstration purposes, only one video file, called custom_hadoop.mp4, is used; you can use any of your own video files to replicate this scenario.

Linking the components

Procedure

1. In the Integration perspective of the Studio, create an empty Job, named FS_video for example, from the Job Designs node in the Repository tree view. For further information about how to create a Job, see Talend Studio User Guide.

2. In the workspace, enter the name of the component to be used and select this component from the list that appears. In this scenario, the components are tMongoDBConnection, tMongoDBGridFSPut,
tMongoDBGridFSPut

tMongoDBGridFSList, tMongoDBGridFSProperties, tFilterColumns, tLogRow, tMongoDBGridFSGet and tMongoDBGridFSDelete.

3. Connect tMongoDBConnection to tMongoDBGridFSPut using the Trigger > On Subjob Ok link.
4. Repeat this operation to connect tMongoDBGridFSPut to tMongoDBGridFSList, tMongoDBGridFSList to tMongoDBGridFSGet, and then tMongoDBGridFSGet to tMongoDBGridFSDelete.
5. Connect tMongoDBGridFSList to tMongoDBGridFSProperties using the Row > Iterate link. This link allows tMongoDBGridFSList to send data to tMongoDBGridFSProperties iteratively.
6. Connect tMongoDBGridFSProperties to tFilterColumns using the Row > Main link.
7. Do the same to connect tFilterColumns to tLogRow.

**Connecting to MongoDB**

**Procedure**

1. Double-click tMongoDBConnection to open its Component view.

2. From the DB version list, select the MongoDB version you are using.
3. In the Server and Port fields, enter the authentication information required for the connection to MongoDB.
   
   If you use the host name of the MongoDB server, ensure that you have added the mapping between this host name and its IP address in the hosts file of the operating system in which the current Job is executed.
4. In the Database field, enter the name of the database hosting GridFS. This database is created on the fly if it does not exist.

**Copying data to MongoDB GridFS**

**Procedure**

1. Double-click tMongoDBGridFSPut to open its Component view.
2. Select the **Use existing connection** check box and from the **Connection** list, select the component in which the MongoDB connection to be used is defined.

3. In the **Bucket** field, enter the bucket to be used to store files in GridFS. In this example, it is `talend_channel/61`.

4. In the **Local folder** field, enter the path, or browse to the folder where the files to be uploaded to GridFS are stored. As explained previously, it is a video file called `custom_hadoop.mp4`.

5. In the **Files** table, add one row by clicking the `[+]` button and in the **Filemask** column, enter `*.mp4` within the double quotation marks. This allows **tMongoDBGridFSPut** to copy all the files with the `.mp4` extension from the local folder you have specified to the bucket to be used in GridFS.

6. Leave the **New name** column empty, that is to say, leave the double quotation marks in this column as is, so as to keep the name of this video unchanged after being copied to GridFS.

**Listing files stored in MongoDB GridFS**

**Iterating on the files**

**Procedure**

1. Double-click **tMongoDBGridFSList** to open its **Component** view.
2. Select the **Use existing connection** check box and from the **Connection** list, select the component in which the MongoDB connection to be used is defined.

3. In the **Bucket** field, enter the bucket in which the files to be listed are stored. In this example, it is `talend_channel/61`.

4. In the **Query** field, enter the query to select the files you want `tMongoDBGridFSList` to iterate on to generate different file lists. In this example, leave the default one in order to iterate on all of the files stored in this `talend_channel/61` bucket.

As explained previously, only one file, `custom_hadoop.mp4`, is expected.

### Extracting file metadata

#### Procedure

1. Double-click `tMongoDBGridFSProperties` to open its **Component** view.

#### tMongoDBGridFSProperties 1

2. Select the **Use existing connection** check box and from the **Connection** list, select the component in which the MongoDB connection to be used is defined.
3. In the **Bucket** field, enter the bucket in which the files to be used are stored. In this example, it is `talend_channel/61`.

4. From the **Query type** list, select the approach you want to use to select the files about which you need to extract the metadata. In this example, select **Filename** to use the `filename` attribute of each GridFS file for query.

5. In the **Filename** field, press **Ctrl + Space** to display the variable list and choose the variable to be used. In this example, select `tMongoDBGridFSList.CURRENT_FILENAME` from the list. Then the expression to use the `CURRENT_FILENAME` variable is automatically added.

   This allows **tMongoDBGridFSProperties** to read each file name returned by **tMongoDBGridFSList**.

### Filtering attributes

#### Procedure

1. Double-click **tFilterColumns** to open its **Component** view.

2. Click the [...] button next to **Edit schema** to open the schema editor.

3. On the left side (the input side), select the column to be used and click the button to move this column to the right side (the output side). In this example, move every column to the right side except the `contentType` column.

   Each column represents a file attribute and the pre-defined schema of **tMongoDBGridFSProperties** automatically contains these columns.

4. Click **OK** to validate these changes and accept the propagation prompted by the pop-up dialog box.
Downloading files from MongoDB GridFS

Procedure

1. Double-click tMongoDBGridFSGet to open its Component view.

   ![tMongoDBGridFSGet Component View]

2. Select the **Use existing connection** check box and from the **Connection** list, select the component in which the MongoDB connection to be used is defined.

3. In the **Bucket** field, enter the bucket in which the files to be retrieved are stored. In this example, it is `talend_channel/61`.

4. In the **Local folder** field, enter the path to the local folder in which you want to store the downloaded files. In this scenario, it is `C:/tmp/output`.

5. Select the **Use Document ID as output filename** check box to rename each downloaded file using the value of its `ObjectId` attribute.

   Since a file in GridFS is distinct by ID rather than by name, it is possible that several files are using the same file name. For this reason, when downloading this kind of files into the same directory without renaming them differently, an exception is returned to alert that the file being downloaded already exists. In order to avoid this error, you can either select the **Overwrite local files** check box to replace the existing one with the latest downloaded file or rename these files on the fly using their IDs. In this example, the strategy of renaming these files is adopted.

Remove files from MongoDB GridFS

Procedure

1. Double-click tMongoDBGridFSDelete to open its Component view.
2. Select the **Use existing connection** check box and from the **Connection** list, select the component in which the MongoDB connection to be used is defined.

3. In the **Bucket** field, enter the bucket in which the files to be deleted are stored. In this example, it is `talend_channel/61`.

4. From the **Query type** list, select the approach you want to use to select the files to be deleted. In this example, select **Filename** to use the `filename` attribute of each GridFS file for query.

5. In the **Filename** field, enter the name of the file to be deleted.

### Executing the Job

**About this task**

Then you can run this Job.

The **tLogRow** component is used to present the execution result of the Job.

**Procedure**

1. If you want to configure the presentation mode on its **Component** view, double-click the **tLogRow** component to open the **Component** view and in the **Mode** area, then, select the **Table (print values in cells of a table)** radio box.

2. Press **F6** to run this Job.

**Results**

Once done, the **Run** view is opened automatically, where the metadata of the video `custom_hadoop.mp4` in GridFS is displayed.

```
Starting job FS_video at 15:00 06-04-2016.
[statistics] connecting to socket on port 3799
[statistics] connected

<table>
<thead>
<tr>
<th>_id</th>
<th>filename</th>
<th>chunkSize</th>
<th>uploadDate</th>
<th>length</th>
<th>md5</th>
</tr>
</thead>
<tbody>
<tr>
<td>57050868d1e6fe7432be242</td>
<td>custom_hadoop.mp4</td>
<td>261120</td>
<td>2016-04-06 15:00:56</td>
<td>66443937</td>
<td>d29ec54715dd09e04a86b5f18bc7854</td>
</tr>
</tbody>
</table>

[statistics] disconnected
```

Job FS_video ended at 15:01 06-04-2016. [exit code=0]
The downloaded file can be found in the directory `C:/tmp/output`, using its ID as its file name.
tMongoDBInput

Retrieves records from a collection in the MongoDB database and transfers them to the following component for display or storage.

`tMongoDBInput` retrieves certain documents from a MongoDB database collection by supplying a query document containing the fields the desired documents should match.

**tMongoDBInput Standard properties**

These properties are used to configure `tMongoDBInput` running in the Standard Job framework.

The `tMongoDBInput` component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>DB Version</strong></td>
<td>List of the database versions. Available when the <code>Use existing connection</code> check box is not selected.</td>
</tr>
<tr>
<td><strong>Use replica set address</strong></td>
<td>Select this check box to show the Replica address table. In the Replica address table, you can define multiple MongoDB database servers for failover. Available when the <code>Use existing connection</code> check box is not selected.</td>
</tr>
<tr>
<td><strong>Server and Port</strong></td>
<td>IP address and listening port of the database server. Available when the <code>Use existing connection</code> or <code>Use replica set address</code> check box is not selected.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Use SSL connection</strong></td>
<td>Select this check box to enable the SSL or TLS encrypted connection. Then you need to use the <code>tSetKeystore</code> component in the same Job to specify the encryption information. For further information about <code>tSetKeystore</code>, see <code>tSetKeystore</code> on page 3532. Note that the SSL connection is available only for the version 2.4 + of MongoDB.</td>
</tr>
<tr>
<td><strong>Set read preference</strong></td>
<td>Select this check box and from the Read preference drop-down list that is displayed, select the member to which you need to direct the read operations.</td>
</tr>
</tbody>
</table>
If you leave this check box clear, the Job uses the default Read preference, that is to say, uses the primary member in a replica set.

For further information, see MongoDB’s documentation about Replication and its Read preferences.

<table>
<thead>
<tr>
<th>Required authentication</th>
<th>Select this check box to enable the database authentication. Among the mechanisms listed on the Authentication mechanism drop-down list, the NEGOTIATE one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using. For details about the other mechanisms in this list, see MongoDB Authentication from the MongoDB documentation.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Set Authentication database</th>
<th>If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name of this Authentication database in the Authentication database field that is displayed. For further information about the MongoDB Authentication database, see User Authentication database.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Username and Password</th>
<th>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. Available when the Required authentication check box is selected. If the security system you have selected from the Authentication mechanism drop-down list is Kerberos, you need to enter the User principal, the Realm and the KDC server fields instead of the Username and the Password fields.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Collection</th>
<th>Name of the collection in the MongoDB database.</th>
</tr>
</thead>
</table>

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  • View schema: choose this option to view the schema only.  
  • Change to built-in property: choose this option to change the schema to Built-in for local changes.  
  • Update repository connection: choose this option to change the schema stored in the repository and |
|-------------------|-----------------------------------------------------------------------------------------------------|
decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.

If a column in the database is a JSON document and you need to read the entire document, put an asterisk (*) in the DB column column, without quotation marks around.

<table>
<thead>
<tr>
<th><strong>Query</strong></th>
<th>Specify the query condition. This field is available only when you have selected <strong>Find query</strong> from the <strong>Query type</strong> drop-down list. For example, type in &quot;(id:4)&quot; to retrieve the record whose id is 4 from the collection specified in the Collection field.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>Different from the query statements required in the MongoDB client software, the query here refers to the contents inside <code>find()</code>, such as the query here <code>{id:4}</code> versus the MongoDB client query <code>db.blog.find({id:4})</code>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Aggregation stages</strong></th>
<th>Create a MongoDB aggregation pipeline by adding the stages you want the documents to pass through so as to obtain aggregated results from these documents. This table is available only when you have selected <strong>Aggregation pipeline query</strong> from the <strong>Query type</strong> drop-down list. Only one stage is allowed per row in this Aggregation stages table and the stages are executed one by one in the order you place them in this table. For example, if you want to aggregate documents about your customers using the <code>$match</code> and the <code>$group</code> stages, you need to add two rows to this Aggregation stages table and define the two stages as follows:</th>
</tr>
</thead>
</table>
| | "{$match : {status : 'A'}}"
"{$group : { _id : '$cust_id',
  total : {$sum : '$amount'}}}" |
| | In this aggregation, the customer documents with status A are selected; then among the selected customers, those using the same customer id are grouped and the values from the amount fields of the same customer are summed up. For a full list of the stages you can use and their related operators, see **Aggregation pipeline operators**. For further information about MongoDB aggregation pipeline, see **Aggregation pipeline**. |

<table>
<thead>
<tr>
<th><strong>Mapping</strong></th>
<th>Each column of the schema defined for this component represents a field of the documents to be read. In this table, you need to specify the parent nodes of these fields, if any. For example, in the document reading as follows</th>
</tr>
</thead>
</table>
The *first* and the *last* fields have *person* as their parent node but the *id* field does not have any parent node. So once completed, this Mapping table should read as follows:

<table>
<thead>
<tr>
<th>Column</th>
<th>Parent node path</th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td></td>
</tr>
<tr>
<td>first</td>
<td>&quot;person&quot;</td>
</tr>
<tr>
<td>last</td>
<td>&quot;person&quot;</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect the log data at the component level.</th>
</tr>
</thead>
<tbody>
<tr>
<td>No query timeout</td>
<td>Select this check box to prevent MongoDB servers from stopping idle cursors at the end of 10-minute inactivity of these cursors. In this situation, an idle cursor will stay open until either the results of this cursor are exhausted or you manually close it using the <code>cursor.close()</code> method.</td>
</tr>
<tr>
<td></td>
<td>A cursor for MongoDB is a pointer to the result set of a query. By default, that is to say, with this check box being clear, a MongoDB server automatically stops idle cursors after a given inactivity period to avoid excess memory use. For further information about MongoDB cursors, see <a href="https://docs.mongodb.org/manual/core/cursors/">https://docs.mongodb.org/manual/core/cursors/</a>.</td>
</tr>
<tr>
<td>Enable external sort</td>
<td>Since the aggregation pipeline stages have a maximum memory use limit (100 megabytes) and a stage exceeding this limit will produce errors, when handling large datasets, select this check box to avoid aggregation stages exceeding this limit.</td>
</tr>
<tr>
<td></td>
<td>For further information about this external sort, see <a href="https://docs.mongodb.org/manual/core/cursors/">Large sort operation with external sort</a>.</td>
</tr>
</tbody>
</table>
Global Variables

**Global Variables**

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

**Usage rule**

As a start component, tMongoDBInput allows you to retrieve records from a collection in the MongoDB database and transfer them to the following component for display or storage.

Scenario: Retrieving data from a collection by advanced queries

This scenario applies only to a Talend solution with Big Data.

In this scenario, advanced MongoDB queries are used to retrieve the post by the author Anderson.

There are such posts in the collection blog of the MongoDB database talend:

```javascript
> db.blog.find()
{ "id" : ObjectId("5226d25e675f26b4a2876888"), "id" : 1, "post" : { "author" : "Andy", "title" : "Open Source Outlook", "keywords" : "Open Source,Talend", "contents" : "Talend, the leader of the open source world..." } }
{ "id" : ObjectId("5226d25e675f26b4a2876889"), "id" : 2, "post" : { "author" : "Andy", "title" : "Data Integration Overview", "keywords" : "Data Integration,Talend", "contents" : "Talend, the leading player in the DI field..." } }
{ "id" : ObjectId("5226d25e675f26b4a287688a"), "id" : 3, "post" : { "author" : "Anderson", "title" : "ELT Overview", "keywords" : "ELT,Talend", "contents" : "Talend, the big name in the ELT circle..." } }
{ "id" : ObjectId("5226d25e675f26b4a287688b"), "id" : 4, "post" : { "author" : "Andy", "title" : "Big Data Bang", "keywords" : "Big Data,Talend", "contents" : "Talend, the driving force for Big Data applications..." } }
```

To insert data into the database, see Scenario 1: Creating a collection and writing data to it on page 2361.
Linking the components

Procedure
1. Drop tMongoDBConnection, tMongoDBClose, tMongoDBInput and tLogRow onto the workspace.
2. Link tMongoDBConnection to tMongoDBInput using the OnSubjobOk trigger.
3. Link tMongoDBInput to tMongoDBClose using the OnSubjobOk trigger.
4. Link tMongoDBInput to tLogRow using a Row > Main connection.

Configuring the components

Procedure
1. Double-click tMongoDBConnection to open its Basic settings view.
2. From the DB Version list, select the MongoDB version you are using.
3. In the Server and Port fields, enter the connection details.
4. In the Database field, enter the name of the MongoDB database.
5. Double-click tMongoDBInput to open its Basic settings view.
6. Select the **Use existing connection** option.

7. In the **Collection** field, enter the name of the collection, namely *blog*.

8. Click the [*...*] button next to **Edit schema** to open the schema editor.

9. Click the [+ ] button to add five columns, namely *id, author, title, keywords* and *contents*, with the type as *Integer* and *String* respectively.

10. Click **OK** to close the editor.

11. The columns now appear in the left part of the **Mapping** area.

12. For columns *author, title, keywords* and *contents*, enter their parent node *post* so that the data can be retrieved from the correct positions.
13. In the **Query** box, enter the advanced query statement to retrieve the posts whose author is Anderson:

```
"{post.author : 'Anderson'}"
```

This statement requires that the sub-node of `post`, the node `author`, should have the value "Anderson".

14. Double-click **tLogRow** to open its **Basic settings** view. Select **Table (print values in cells of a table)** for a better display of the results.

**Executing the Job**

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to run the Job.

```
<table>
<thead>
<tr>
<th>id</th>
<th>author</th>
<th>title</th>
<th>keywords</th>
<th>contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Anderson</td>
<td>ELT Overview</td>
<td>ELT, Talend</td>
<td>the big name in the ELT circle...</td>
</tr>
</tbody>
</table>
```

As shown above, the post by Anderson is retrieved.

**Related scenarios**

For related scenarios, see:

- **Scenario 1: Creating a collection and writing data to it** on page 2361
- **Scenario: Using MongoDB functions to create a collection and write data to it** on page 2376
tMongoDBOutput

Executes the action defined on the collection in the MongoDB database. tMongoDBOutput inserts, updates, upserts or deletes documents in a MongoDB database collection based on the incoming flow from the preceding component in the Job.

### tMongoDBOutput Standard properties

These properties are used to configure tMongoDBOutput running in the Standard Job framework. The Standard tMongoDBOutput component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

#### Basic settings

<table>
<thead>
<tr>
<th>Use existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Version</td>
<td>List of the database versions. Available when the Use existing connection check box is not selected.</td>
</tr>
<tr>
<td>Use replica set address</td>
<td>Select this check box to show the Replica address table. In the Replica address table, you can define multiple MongoDB database servers for failover. Available when the Use existing connection check box is not selected.</td>
</tr>
<tr>
<td>Server and Port</td>
<td>IP address and listening port of the database server. Available when the Use existing connection or Use replica set address check box is not selected.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Use SSL connection</td>
<td>Select this check box to enable the SSL or TLS encrypted connection. Then you need to use the tSetKeystore component in the same Job to specify the encryption information. For further information about tSetKeystore, see tSetKeystore on page 3532. Note that the SSL connection is available only for the version 2.4 + of MongoDB.</td>
</tr>
<tr>
<td>Set write concern</td>
<td>Select this check box to set the level of acknowledgement requested from for write operations. Then you need to select the level of this operation.</td>
</tr>
</tbody>
</table>
For further information, see the related MongoDB documentation on [http://docs.mongodb.org/manual/core/write-concern/](http://docs.mongodb.org/manual/core/write-concern/).

### Bulk write

Select this check box to insert, update or remove data in bulk. Note this feature is available only when the version of MongoDB you are using is 2.6+.

Then you need to select **Ordered** or **Unordered** to define how the MongoDB database processes the data sent by the Studio.

- If you select **Ordered**, MongoDB processes the queries sequentially.
- If you select **Unordered**, MongoDB optimizes the bulk write operations without keeping the order in which the individual operations were inserted in the bulk write.

In the **Bulk write size** field, enter the size of each query group to be processed by MongoDB. In the documentation of MongoDB, some restrictions and expected behaviors as to this size are explained. You can find the details on [http://docs.mongodb.org/manual/core/bulk-write-operations/](http://docs.mongodb.org/manual/core/bulk-write-operations/).

### Required authentication

Select this check box to enable the database authentication.

Among the mechanisms listed on the **Authentication mechanism** drop-down list, the **NEGOTIATE** one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using.

For details about the other mechanisms in this list, see **MongoDB Authentication** from the MongoDB documentation.

### Set Authentication database

If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name of this Authentication database in the **Authentication database** field that is displayed.

For further information about the MongoDB Authentication database, see **User Authentication database**.

### Username and Password

DB user authentication data.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

Available when the **Required authentication** check box is selected.

If the security system you have selected from the **Authentication mechanism** drop-down list is Kerberos, you need to enter the **User principal**, the **Realm** and
<table>
<thead>
<tr>
<th><strong>Collection</strong></th>
<th>Name of the collection in the MongoDB database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Drop collection if exist</strong></td>
<td>Select this check box to drop the collection if it already exists.</td>
</tr>
<tr>
<td><strong>Action on data</strong></td>
<td>The following operations are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Insert</strong>: insert documents.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Set</strong>: modifies the existing fields of an existing document and appends a field if it does not exist in this document. If you need to apply this action on all the documents in the collection to be used, select the <strong>Update all document</strong> check box that is displayed; otherwise, only the first document is updated.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update</strong>: replaces the existing documents with the incoming data but keeps the technical ID of these documents.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Upsert</strong>: inserts a document if it does not exist otherwise it applies the same rules as <strong>Update</strong>.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Upsert with set</strong>: inserts a document if it does not exist otherwise it applies the same rules as <strong>Set</strong>. If you need to apply this action on all the documents in the collection to be used, select the <strong>Update all document</strong> check box that is displayed; otherwise, only the first document is updated.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Delete</strong>: delete documents.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields. Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window. Click <strong>Sync columns</strong> to retrieve the schema from the previous component connected in the Job.</td>
</tr>
</tbody>
</table>
**Built-In:** You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository:** You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

**Mapping**

Each column of the schema defined for this component represents a field of the documents to be read. In this table, you need to specify the parent nodes of these fields, if any.

For example, in the document reading as follows

```json
{  
    _id: ObjectId("5099803df3f4948bd2f98391"),
    person: { first: "Joe", last: "Walker" }
}
```

The `first` and the `last` fields have `person` as their parent node but the `_id` field does not have any parent node.

So once completed, this **Mapping** table should read as follows:

<table>
<thead>
<tr>
<th>Column</th>
<th>Parent node path</th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td></td>
</tr>
<tr>
<td>first</td>
<td>&quot;person&quot;</td>
</tr>
<tr>
<td>last</td>
<td>&quot;person&quot;</td>
</tr>
</tbody>
</table>

Not available when the **Generate JSON Document** check box is selected in **Advanced settings**.

**Die on error**

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

**Advanced settings**

**Generate JSON Document**

Select this check box for JSON configuration:

**Configure JSON Tree:** click the [...] button to open the interface for JSON tree configuration. For more information, see **Configuring a JSON Tree** on page 4040.

**Group by:** click the [+] button to add lines and choose the input columns for grouping the records.

**Remove root node:** select this check box to remove the root node.
Data node and Query node (available for update and upsert actions): type in the name of data node and query node configured on the JSON tree.

**Warning:**
These nodes are mandatory for update and upsert actions. They are intended to enable the update and upsert actions though will not be stored in the database.

<table>
<thead>
<tr>
<th>No query timeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to prevent MongoDB servers from stopping idle cursors at the end of 10-minute inactivity of these cursors. In this situation, an idle cursor will stay open until either the results of this cursor are exhausted or you manually close it using the cursor.close() method. A cursor for MongoDB is a pointer to the result set of a query. By default, that is to say, with this check box being clear, a MongoDB server automatically stops idle cursors after a given inactivity period to avoid excess memory use. For further information about MongoDB cursors, see <a href="https://docs.mongodb.org/manual/core/cursors/">https://docs.mongodb.org/manual/core/cursors/</a>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to collect the log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
</tr>
</thead>
</table>
| **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tMongoDBOutput</strong> executes the action defined on the collection in the MongoDB database based on the flow incoming from the preceding component in the Job.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
</tr>
</tbody>
</table>
• The "multi" parameter, which allows to update multiple documents at a time, is not supported. Therefore, if two documents have the same key, the first is always updated, but the second never will.

• For the update operation, the key cannot be a JSON array.

Scenario 1: Creating a collection and writing data to it

This scenario applies only to a Talend solution with Big Data.

This scenario creates the collection blog and writes post data to it.

Linking the components

Procedure

1. Drop tMongoDBConnection, tFixedFlowInput, tMongoDBOutput, tMongoDBClose, tMongoDBInput and tLogRow onto the workspace.
2. Rename tFixedFlowInput as blog_post_data, tMongoDBOutput as write_data_to_collection, tMongoDBInput as read_data_from_collection and tLogRow as show_data_from_collection.
3. Link tMongoDBConnection to tFixedFlowInput using the OnSubjobOk trigger.
4. Link tFixedFlowInput to tMongoDBOutput using a Row > Main connection.
5. Link tFixedFlowInput to tMongoDBInput using the OnSubjobOk trigger.
7. Link tMongoDBInput to tLogRow using a Row > Main connection.

Configuring the components

Procedure

1. Double-click tMongoDBConnection to open its Basic settings view.
2. From the **DB Version** list, select the MongoDB version you are using.

3. In the **Server** and **Port** fields, enter the connection details.
   
   In the **Database** field, enter the name of the MongoDB database.

4. Double-click **tFixedFlowInput** to open its **Basic settings** view.

Select **Use Inline Content (delimited file)** in the **Mode** area.

In the **Content** field, enter the data to write to the MongoDB database, for example:

```
1;Andy;Open Source Outlook;Open Source,Talend;Talend, the leader of the open source world...
3;Andy;ELT Overview;ELT,Talend;Talend, the big name in the ELT circle...
2;Andy;Data Integration Overview;Data Integration,Talend;Talend, the leading player in the DI field...
```

5. Double-click **tMongoDBOutput** to open its **Basic settings** view.
Select the **Use existing connection** and **Drop collection if exist** check boxes.

In the **Collection** field, enter the name of the collection, namely *blog*.

6. Click the [...] button next to **Edit schema** to open the schema editor.

7. Click the [+] button to add five columns in the right part, namely *id, author, title, keywords and contents*, with the type as *Integer* and *String* respectively.

   - Click **→** to copy all the columns to the input table.
   - Click **Ok** to close the editor.

8. The columns now appear in the left part of the **Mapping** area.

   - For columns *author, title, keywords and contents*, enter their parent node *post*. By doing so, those nodes reside under the node *post* in the MongoDB collection.

9. Double-click **tMongoDBInput** to open its **Basic settings** view.
Select the **Use existing connection** check box.

In the **Collection** field, enter the name of the collection, namely *blog*.

10. Click the [...] button next to **Edit schema** to open the schema editor.

11. Click the [+] button to add five columns, namely *id*, *author*, *title*, *keywords* and *contents*, with the type as *Integer* and *String* respectively.

   Click **OK** to close the editor.

12. The columns now appear in the left part of the **Mapping** area.
For columns author, title, keywords and contents, enter their parent node post so that the data can be retrieved from the correct positions.

13. In the Sort by area, click the [+] button to add one line and enter id under Column. Select asc from the Order asc or desc? column to the right of the id column. This way, the retrieved records will appear in ascending order of the id column.

**Executing the Job**

**Procedure**

1. Press Ctrl+S to save the Job.
2. Press F6 to run the Job.

```
    | statistics] connecting to socket on port 27017
    | show_data_from_collection
    | id  | author     | title               | keywords       | contents                                 |
    +----+------------+---------------------+-------------------+------------------------------------------|
    | 1   | Andy       | Open Source Outlook | Open Source, Talend | Talend, the leader of the open source world... |
    | 2   | Andy       | Data Integration Overview | Data Integration, Talend | Talend, the leading player in the DI field... |
    | 3   | Andy       | ELT Overview | ELT, Talend | Talend, the big name in the ELT circle... |

    | statistics] disconnected
```

3. Switch to the database talend and read data from the collection blog in the MongoDB command line client. You can find that author, title, keywords and contents all reside under the node post. Meanwhile, the records are stored in the same order as the source input.

**Scenario 2: Upserting records in a collection**

This scenario applies only to a Talend solution with Big Data.

This scenario upserts the collection blog as an existing record has its author changed and a new record is added. Before the upsert, the collection blog looks like:

```
1;Andy;Open Source Outlook;Open Source,Talend;Talend, the leader of the open source world...
2;Andy;Data Integration Overview;Data Integration,Talend;Talend, the leading player in the DI field...
3;Andy;ELT Overview;ELT,Talend;Talend, the big name in the ELT circle...
```
Such records can be inserted to the database following the instructions of Scenario 1: Creating a collection and writing data to it on page 2361.

**Linking the components**

**Procedure**

1. Drop tMongoDBConnection, tFixedFlowInput, tMongoDBOutput, tMongoDBClose, tMongoDBInput and tLogRow from the Palette onto the design workspace.
2. Rename tFixedFlowInput as blog_post_data, tMongoDBOutput as write_data_to_collection, tMongoDBInput as read_data_from_collection and tLogRow as show_data_from_collection.
3. Link tMongoDBConnection to tFixedFlowInput using the OnSubjobOk trigger.
4. Link tFixedFlowInput to tMongoDBOutput using a Row > Main connection.
5. Link tFixedFlowInput to tMongoDBInput using the OnSubjobOk trigger.
7. Link tMongoDBInput to tLogRow using a Row > Main connection.

**Configuring the components**

**Procedure**

1. Double-click tMongoDBConnection to open its Basic settings view.

   ![tMongoDBConnection](image)

2. From the DB Version list, select the MongoDB version you are using.
3. In the Server and Port fields, enter the connection details.
In the **Database** field, enter the name of the MongoDB database.

4. Double-click **tFixedFlowInput** to open its **Basic settings** view.

   ![Basic settings view](image)

   Select **Use Inline Content (delimited file)** in the **Mode** area.

   In the **Content** field, enter the data for upserting the MongoDB database, for example:

   ```
   1;Andy;Open Source Outlook;Open Source,Talend,Talend, the leader of the open source world...
   2;Andy;Data Integration Overview;Data Integration,Talend,Talend, the leading player in the DI field...
   3;Anderson;ELT Overview;ELT,Talend,Talend, the big name in the ELT circle...
   4;Andy;Big Data Bang;Big Data,Talend,Talend, the driving force for Big Data applications...
   ```

   As shown above, the 3rd record has its author changed and the 4th record is new.

5. Double-click **tMongoDBOutput** to open its **Basic settings** view.

   ![Basic settings view](image)

   Select the **Use existing connection** and **Die on error** check boxes.

   In the **Collection** field, enter the name of the collection, namely **blog**.

   Select **Upsert** from the **Action on data** list.
6. Click the [...] button next to **Edit schema** to open the schema editor.

![Schema of write_data_to_collection](image)

7. Click the [+] button to add five columns in the right part, namely *id*, *author*, *title*, *keywords* and *contents*, with the type as *Integer* and *String* respectively.

   Click ![copy button](image) to copy all the columns to the input table.

   Click **Ok** to close the editor.

8. In the **Advanced Settings** view, select the **Generate JSON Document** check box.

   Select the **Remove root node** check box.

   In the **Data node** and **Query node** fields, enter "*data*" and "*query*".

![write_data_to_collection(tMongoDBOutput_4)](image)

9. Click the [...] button next to **Configure JSON Tree** to open the configuration interface.
10. Right-click the node `rootTag` and select **Add Sub-element** from the contextual menu. In the dialog box that appears, type in `data` for the **Data node**:

![Input element's label](image1)

Click **OK** to close the window.

Repeat this operation to define `query` as the **Query node**.

Right-click the node `data` and select **Set As Loop Element** from the contextual menu.

**Warning:**

These nodes are mandatory for update and upsert actions. They are intended to enable the update and upsert actions though will not be stored in the database.

11. Select all the columns under the **Schema list** and drop them to the `data` node.

In the window that appears, select **Create as sub-element of target node**.
Click OK to close the window.
Repeat this operation to drop the id column from the Schema list under the Query node.

12. Right-click the node id under data and select Add Attribute from the contextual menu.
In the dialog box that appears, type in type as the attribute name:

Click OK to close the window.
Right-click the node @type under id and select Set A Fix Value from the contextual menu.
In the dialog box that appears, type in integer as the attribute value, ensuring the id values are stored as integers in the database.

Click OK to close the window.
Repeat this operation to set this attribute for the id node under Query.
Click OK to close the JSON Tree configuration interface.

Select the **Use existing connection** check box.

In the **Collection** field, enter the name of the collection, namely *blog*.

Click the [...] button next to **Edit schema** to open the schema editor.

Click the [*+] button to add five columns, namely *id, author, title, keywords and contents*, with the type as *Integer* and *String* respectively.

Click **OK** to close the editor.

The columns now appear in the left part of the **Mapping** area.

For columns *author, title, keywords and contents*, enter their parent node *post* so that the data can be retrieved from the correct positions.

14. Double-click **tLogRow** to open its **Basic settings** view.
In the **Mode** area, select **Table (print values in cells of a table)** for better display.

### Executing the Job

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to run the Job.

```
<table>
<thead>
<tr>
<th>id</th>
<th>author</th>
<th>title</th>
<th>keywords</th>
<th>contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Andy</td>
<td>Open Source Outlook</td>
<td>Open Source Talend</td>
<td>Talend: the leader of the open source world...</td>
</tr>
<tr>
<td>2</td>
<td>Andy</td>
<td>Data Integration Overview</td>
<td>Data Integration Talend</td>
<td>Talend: the leading player in the DI field...</td>
</tr>
<tr>
<td>3</td>
<td>Anderson</td>
<td>ELT Overview</td>
<td>ELT Talend</td>
<td>Talend: the big name in the ELT circle...</td>
</tr>
<tr>
<td>4</td>
<td>Andy</td>
<td>Big Data Magic</td>
<td>Big Data Talend</td>
<td>Talend: the driving force for Big Data applications...</td>
</tr>
</tbody>
</table>
```

As shown above, the 3rd record has its author updated and the 4th record is inserted.
tMongoDBRow

Executes the commands and functions of the MongoDB database.

### tMongoDBRow Standard properties

These properties are used to configure tMongoDBRow running in the Standard Job framework.

The Standard tMongoDBRow component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

#### Basic settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>DB Version</strong></td>
<td>List of the database versions. Available when the Use existing connection check box is not selected.</td>
</tr>
<tr>
<td><strong>Use replica set address</strong></td>
<td>Select this check box to show the Replica address table. In the Replica address table, you can define multiple MongoDB database servers for failover. Available when the Use existing connection check box is not selected.</td>
</tr>
<tr>
<td><strong>Server and Port</strong></td>
<td>Address and listening port of the database server. Available when the Use existing connection or Use replica set address check box is not selected.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Use SSL connection</strong></td>
<td>Select this check box to enable the SSL or TLS encrypted connection. Then you need to use the tSetKeystore component in the same Job to specify the encryption information. For further information about tSetKeystore, see tSetKeystore on page 3532. Note that the SSL connection is available only for the version 2.4 + of MongoDB.</td>
</tr>
<tr>
<td><strong>Required authentication</strong></td>
<td>Select this check box to enable the database authentication. Among the mechanisms listed on the Authentication mechanism drop-down list, the NEGOTIATE one is recommended if you are not using Kerberos, because it automatically select the authentication mechanism the most adapted to the MongoDB version you are using.</td>
</tr>
</tbody>
</table>
For details about the other mechanisms in this list, see [MongoDB Authentication](https://docs.mongodb.com/manual/reference/auth/) from the MongoDB documentation.

### Set Authentication database
If the username to be used to connect to MongoDB has been created in a specific Authentication database of MongoDB, select this check box to enter the name of this Authentication database in the **Authentication database** field that is displayed.

For further information about the MongoDB Authentication database, see [User Authentication database](https://docs.mongodb.com/manual/reference/auth/).

### Username and Password
DB user authentication data.

To enter the password, click the `...` button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

Available when the **Required authentication** check box is selected.

If the security system you have selected from the **Authentication mechanism** drop-down list is Kerberos, you need to enter the **User principal**, the **Realm** and the **KDC server** fields instead of the **Username** and the **Password** fields.

### Schema and Edit Schema
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word *line* when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.

Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

### Execute command
Select this check box to enter MongoDB commands in the **Command** field for execution.

- **Command**: in this field, enter the command to be executed, if this command contains one single variable.
For example, if you need to construct the command

```json
{"isMaster": 1}
```

You need simply enter `isMaster` within quotation marks.

- **Construct command from keys and values**: if the command to be executed contains multiple variables, select this check box and in the **Command keys and values** table, add the variables and their respective values to be used.

  For example, if you need to construct the following command

  ```json
  { renameCollection : 
    "<source_namespace>" ,
    to : "<target_namespace>" ,
    dropTarget : < true | false > }
  ```

  You need to add three rows to the **Command keys and values** table and enter one variable-value pair to each row within quotation marks:

  ```json
  "renameCollection"
  "old_name"
  "to"
  "new_name"
  "dropTarget"            "false"
  ```

- **Construct command from a JSON string**: if you want to directly enter the command to be used, select this check box and enter this command in the **JSON string command** field that is displayed. Only one command is allowed per **tMongoDBRow**.

  For example:

  ```json
  "{createIndexes: 'restaurants',
  indexes : [{key : {
  restaurant_id: 1}, name:
  'id_index_2', unique: true}]}"
  ```

  Note that you must use single quotation marks to surround the string values used in the command and double quotation marks to surround the command itself.

  For further information about the MongoDB commands you can use in this field, see [https://docs.mongodb.org/manual/reference/command/](https://docs.mongodb.org/manual/reference/command/).

<table>
<thead>
<tr>
<th>Function</th>
<th>Enter MongoDB functions in the <strong>Function</strong> field for execution. Not available when the <strong>Execute command</strong> check box is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters value</td>
<td>Click the [*] button to add lines as needed and then define the parameter values in the form of variables or <strong>tMongoDBRow</strong>.</td>
</tr>
</tbody>
</table>
constant values, for example `row1.author` or "Andy". Note that the parameter values correspond to the parameters defined in the Function field, in the same order.

Not available when the **Execute command** check box is selected.

**Die on error**

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect the log data at the component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see **Talend Studio User Guide**. |

**Usage**

| Usage rule | **tMongoDBRow** allows you to manipulate the MongoDB database through the MongoDB commands and functions. |

**Scenario: Using MongoDB functions to create a collection and write data to it**

This scenario applies only to a Talend solution with Big Data.

This scenario creates the collection *blog* and writes post data to it through the MongoDB functions.

**Linking the components**

**Procedure**

1. Drop **tMongoDBConnection**, **tFixedFlowInput**, **tMongoDBRow**, **tMongoDBClose**, **tMongoDBInput** and **tLogRow** onto the workspace.
2. Rename `tFixedFlowInput` as `blog_post_data`, `tMongoDBRow` as `write_data_to_collection`, `tMongoDBInput` as `read_data_from_collection` and `tLogRow` as `show_data_from_collection`.

3. Link `tMongoDBConnection` to `tFixedFlowInput` using the OnSubjobOk trigger.

4. Link `tFixedFlowInput` to `tMongoDBRow` using a Row > Main connection.

5. Link `tFixedFlowInput` to `tMongoDBInput` using the OnSubjobOk trigger.


7. Link `tMongoDBInput` to `tLogRow` using a Row > Main connection.

### Configuring the components

**Procedure**

1. Double-click `tMongoDBConnection` to open its Basic settings view.

![tMongoDBConnection_1](image)

2. From the DB Version list, select the MongoDB version you are using.

3. In the Server and Port fields, enter the connection details.

   In the Database field, enter the name of the MongoDB database.

4. Double-click `tFixedFlowInput` to open its Basic settings view.
Select **Use Inline Content (delimited file)** in the **Mode** area.

In the **Content** field, enter the data to write to the MongoDB database, for example:

Andy;Open Source Outlook;Open Source,Talend,Talend, the leader of the open source world...
Andy;Data Integration Overview;Data Integration,Talend,Talend, the leading player in the DI field...
Andy;ELT Overview;ELT,Talend,Talend, the big name in the ELT circle...

5. Double-click **tMongoDBRow** to open its **Basic settings** view.
Select the **Use existing connection** check box.

In the **Function** field, enter the MongoDB function to create the collection `blog` and insert data to it:

```javascript
"function(author,title,keywords,contents){
    return db.blog.save(
        {author:author,title:title,keywords:keywords,contents:contents}
    );
}
"
```

6. Click the [...] button next to **Edit schema** to open the schema editor.
7. Click the [+ ] button to add four columns in the right part, namely author, title, keywords and contents, with the type of String.

Click to copy all the columns to the input table.

Click OK to close the editor.

8. In the Parameters value table, click the [+ ] button to add four lines and enter the values in sequence: row1.author, row1.title, row1.keywords and row1.contents. By doing so, data of row1 will be transferred to the parameters defined in the function.


Select the Use existing connection check box.

In the Collection field, enter the name of the collection, namely blog.

10. Click the [...] button next to Edit schema to open the schema editor.

11. Click the [+ ] button to add four columns, namely author, title, keywords and contents, with the type as String.
Click **OK** to close the editor.

12. Double-click **tLogRow** to open its **Basic settings** view.

In the **Mode** area, select **Table (print values in cells of a table)** for better display.

**Executing the Job**

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to run the Job.

<table>
<thead>
<tr>
<th>author</th>
<th>title</th>
<th>keywords</th>
<th>contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andy</td>
<td>Open Source Outlook</td>
<td>Open Source Talend</td>
<td>Talend, the leader of the open source world...</td>
</tr>
<tr>
<td>Andy</td>
<td>Data Integration Overview</td>
<td>Data Integration, Talend</td>
<td>Talend, the leading player in the DI field...</td>
</tr>
<tr>
<td>Andy</td>
<td>ELT Overview</td>
<td>ELT, Talend</td>
<td>Talend, the big name in the ELT circle...</td>
</tr>
</tbody>
</table>
tMSAXInput

Extracts data from a Microsoft AX server based on a query.

**tMSAXInput Standard properties**

These properties are used to configure tMSAXInput running in the Standard Job framework.

The Standard tMSAXInput component belongs to the Business family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the Repository file where properties are stored. The fields that come after are pre-filled in using the fetched data.</td>
</tr>
</tbody>
</table>

**Version**

Select the type of the business connector to be used.

The type may be:

- Dynamics AX 4.0
- Dynamics AX 2012

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</th>
</tr>
</thead>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
</tbody>
</table>
For further information about variables, see *Talend Studio User Guide*.

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is usually used as a start component. An output component is required.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td><strong>.NET Business Connector Assembly Path</strong></td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td><strong>Dynamics AX 2012 type only.</strong></td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td><strong>Dynamics AX 2012 type only.</strong></td>
</tr>
<tr>
<td><strong>AOS Server Instance</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Domain</strong></td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td><strong>Dynamics AX 2012 type only.</strong></td>
</tr>
<tr>
<td><strong>User and Password</strong></td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td><strong>Dynamics AX 2012 type only.</strong></td>
</tr>
<tr>
<td><strong>Company</strong></td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td><strong>Dynamics AX 2012 type only.</strong></td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td><strong>Dynamics AX 2012 type only.</strong></td>
</tr>
<tr>
<td><strong>Configuration File</strong></td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td><strong>Dynamics AX 2012 type only.</strong></td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Table Name</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tMSAXOutput

Writes data in a Microsoft AX server.

**tMSAXOutput Standard properties**

These properties are used to configure tMSAXOutput running in the Standard Job framework.

The Standard tMSAXOutput component belongs to the Business family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the Repository file where properties are stored. The fields that come after are pre-filled in using the fetched data.</td>
</tr>
</tbody>
</table>

**Version**

Select the type of the business connector to be used. The type may be:

- Dynamics AX 4.0
- Dynamics AX 2012

### Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

### Global Variables

**Global Variables**

- **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.
- **NB_LINE_UPDATED**: the number of rows updated. This is an After variable and it returns an integer.
- **NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.
- **NB_LINE_DELETED**: the number of rows deleted. This is an After variable and it returns an integer.
- **NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions...
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as an output component. An Input component is required.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>.NET Business Connector Assembly Path</strong></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>Host</strong></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>Port</strong></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>AOS Server Instance</strong></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>Domain</strong></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>User and Password</strong></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>Company</strong></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>Language</strong></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>Configuration File</strong></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>Table Name</strong></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>Action on data</strong></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>Schema and Edit Schema</strong></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>Die on error</strong></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /> <strong>Dynamics AX 2012</strong> type only.</td>
<td><strong>Additional Columns</strong></td>
</tr>
</tbody>
</table>
Scenario 1: Inserting data in a defined table in a MicrosoftAX server

Warning:
Before being able to use this component, make sure that you install and launch the MicrosoftAX server correctly.

This Java scenario describes a two-component Job that uses tMSAXOutput to insert four columns in a defined table in a MicrosoftAX server after it alters values in one of the inserted columns.

Setting up the Job

Procedure

1. Drop tFixedFlowInput and tMSAXOutput from the Palette to the design workspace.
2. Connect the two components together using a Row > Main connection.

Configuring tFixedFlowInput

Procedure

1. Double-click tFixedFlowInput to display its Basic settings view and define the component properties.
2. Set Schema type to Built-in and click the three-dot button next to Edit schema to display a dialog box where you can define the input schema.
3. Click the plus button and add the input schema columns, three in this example: name, city and street.
4. Click **OK** to close the dialog box and accept propagating the changes when prompted by the system.

The three schema columns display automatically in the **Values** list.

5. Click in the **Value** column and enter a value for each of the input columns.

**Configuring tMSAXOutput**

**Procedure**

1. Double-click **tMSAXOutput** to open its **Basic settings** view and define the component properties.

2. Set **Property type** to **Built-in**.

3. In the **Host** field, type in the IP address of the MicrosoftAX server and type in the domain name on which the MicrosoftAX server is hosted in the **Domain** field.

4. Enter your username and password for the server in the corresponding fields and enter the name of the table you want to write data in the **Table Name** field, ADDRESS in this example.

5. In the **Action on data** list, select the action you want to carry on, **Insert** in this example.

6. Click **Sync columns** to retrieve the schema from the preceding component.

   In this example, we want to retrieve the three input columns: *name*, *city* and *street* and write the data included in the three input columns in the MicrosoftAX server without any changes.

   If needed, click the three-dot button next to **Edit Schema** to verify the retrieved schema.

7. In the **Additional columns** list, click the plus button to add one line where you can define parameters for the new column to add to the row you want to write in the ADDRESS table.

8. Set a name, a data type, a position and a reference column in the corresponding columns for the line you added.

   In this example, we want to add a new column we call "address" after the *street* column.

9. Click in the **Local expression** column and press **Ctrl + space** on your keyboard to open the context variable list and select: `StringHandling.UPCASE(row2.city)+"-"+row2.street`. This expression will write the city name initially capped followed by the street name to form the address of Bryant park. Thus the address column in this example will contain the string: *New York-Midtown Manhattan*. 
Job execution

Procedure

Save your Job and press F6 to execute it.

`tMSAXOutput` inserts in the `ADDRESS` table in the MicrosoftAX server a row that holds the three input columns, `name`, `city` and `street` in addition to the new `address` column that combines the city name and the street name.

Scenario 2: Deleting data from a defined table in a MicrosoftAX server

Warning:

Before being able to use this component, make sure that you install and launch the MicrosoftAX server correctly.

This Java scenario describes a two-component Job that uses `tMSAXOutput` to delete from a defined table in a MicrosoftAX server all rows that do not match the data included in a key column.

In this example, the input schema we use is an `address` column that holds the following data: *New York-Midtown Manhattan*. We want to delete from the MicrosoftAX server all addresses that are not identical with this one.

Setting up the Job

Procedure

1. Drop `tFixedFlowInput` and `tMSAXOutput` from the Palette to the design workspace.
2. Connect the two components together using a Row > Main connection.

Configuring `tFixedFlowInput`

Procedure

1. Double-click `tFixedFlowInput` to display its Basic settings view and define the component properties.
2. Set Schema type to Built-in and click the three-dot button next to Edit schema to display a dialog box where you can define the input schema.

3. Click the plus button and add the input schema columns, address in this example.

4. Click OK to close the dialog box. The schema column displays automatically in the Values list.

5. Click in the Value column and enter a value for the input column.

**Setting up the connection to the MicrosoftAX server**

**Procedure**

1. Double-click tMSAXOutput to open its Basic settings view and define the component properties.

2. Set Property type to Built-in.

3. In the Host field, type in the IP address of the MicrosoftAX server.

4. In the Domain field, type in the domain name on which the MicrosoftAX server is hosted.

5. Enter your username and password for the server in the corresponding fields.

6. In the Table Name field, enter the name of the table you want to delete data from, ADDRESS in this example.
Defining the action on data

Procedure

1. In the Action on data list, select the action you want to carry on, **Delete** in this example.
2. Click **Sync columns** to retrieve the schema from the preceding component. In this example, we want to retrieve the input column: *address*.
3. Click the three-dot button next to **Edit Schema** to open a dialog box where you can verify the retrieved schema.

4. In the output schema, select the Key check box next to the column name you want to define as a key column, and then click **OK** to validate your changes and close the dialog box.

   ![Schema dialog box](image)

   **Note:**
   When you select **Delete** as an action on data, you must always define the Reference column as a key column in order for **tMSAXOutput** to delete rows based on this key column.

5. In the Additional columns list, click the plus button to add one line and define the parameters the component will use as basis for the delete operation.
6. Set a name, an operator, a data type, a local expression, a position and a reference column in the corresponding columns for the line you added.

   In this example, we want to delete from the *ADDRESS* table in the MicrosoftAX server all rows in which the address column is not equal to the address in the key *address* column and that reads as the following: New York-Midtown Manhattan.

   **Note:**
   When you select **Delete** as an action on data, you must always set Position to Replace. Otherwise, all settings in the Additional columns will not be taken into account when executing your Job.

Job execution

Procedure

Save your Job and press **F6** to execute it.

**tMSAXOutput** deletes from the *ADDRESS* table in the MicrosoftAX server all rows where the address string is not equal to the address in the key column.
tMsgBox

Opens a dialog box with an OK button requiring action from the user.
tMsgBox is a graphical break in the Job execution progress.

**tMsgBox Standard properties**

These properties are used to configure tMsgBox running in the Standard Job framework.
The Standard tMsgBox component belongs to the Misc family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Title</th>
<th>Text entered shows on the title bar of the dialog box created.</th>
</tr>
</thead>
</table>
| Buttons | Listbox of buttons you want to include in the dialog box. The button combinations are restricted and cannot be changed.  
The **Question** button displays the **Mask Answer** check box. Select this check box if you want to mask the answer you type in the pop-up window that opens when you run the Job. |
| Icon | Icon shows on the title bar of the dialog box. |
| Message | Free text to display as message on the dialog box. Text can be dynamic (for example: retrieve and show a file name). |

**Global Variables**

| Global Variables | RESULT: the return value of the component. This is an After variable and it returns a string.  
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |
|-------------------|---------------------------------------------------------------|

2391
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as intermediate step in a data flow or as a start or an end object in the Job flowchart. It can be connected to the next/previous component using either a Row or Iterate link.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Scenario: 'Hello world!' type test

The following scenario creates a single-component Job, where tMsgBox is used to display the pid (process id) in place of the traditional “Hello World!” message.

Setting up the Job

Procedure

1. Drop a tMsgBox component from the Palette to the design workspace.

2. Define the dialog box display properties:

   - **Title**: "Talend Open Studio"
   - **Buttons**: OK
   - **Icon**: Icon Information
   - **Message**: "Current date is: "+(TalendDate.getCurrentDate())

   ‘Title’ is the message box title, it can be any variable.

3. In the Message field, enter "Current date is: " between double quotation marks. Then click CTRL+Space to display the autocompletion list and select the following system routine, TalendDate.getCurrentDate. Put brackets around this routine.

Executing the Job

Procedure

- Switch to the Run tab to execute the Job defined.

Results

The Message box displays the message and requires the user to click **OK** to go to the next component or end the Job.
After the user clicked **OK**, the **Run** log is updated accordingly.

Related topic: see *Talend Studio User Guide*. 
tMSSqlBulkExec

 Offers gains in performance while executing the Insert operations to a Microsoft SQL Server database.

The tMSSqlOutputBulk and tMSSqlBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tMSSqlOutputBulkExec component, detailed in a separate section. The advantage of using a two step process is that the data can be transformed before it is loaded in the database.

**tMSSqlBulkExec Standard properties**

These properties are used to configure tMSSqlBulkExec running in the Standard Job framework.

The Standard tMSSqlBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data is stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**JDBC Provider**

Select the provider of the JDBC driver to be used, either Microsoft (recommended) or **Open source JTDS**.

When using this component with Datasource in Talend Runtime, you need to use the **Open source JTDS** driver.

Note that when Microsoft is selected, you need to download the Microsoft JDBC Driver for SQL Server on Microsoft Download Center, unpack the downloaded zip file, choose a jar in the unzipped folder based on your JRE version, rename the jar to *mssql-jdbc.jar* and install it manually. For more information about choosing the jar, see the System Requirements information on Microsoft Download Center.

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>
### Note:

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](#).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data.  To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time and that the table must exist for the insert operation to succeed.</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td>On the table defined, you can perform one of the following operations:  <strong>None</strong>: No operation is carried out.  <strong>Drop and create table</strong>: The table is removed and created again.  <strong>Create table</strong>: The table does not exist and gets created.  <strong>Create table if not exists</strong>: The table is created if it does not exist.  <strong>Drop table if exists and create</strong>: The table is removed if it already exists and created again.  <strong>Clear table</strong>: The table content is deleted.  <strong>Truncate table</strong>: The table content is deleted. You do not have the possibility to rollback the operation.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the</td>
</tr>
</tbody>
</table>
next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

| **Built-In** | You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*. |
| **Repository** | You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*. When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)). |

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-In** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

**Remote File Name**

Name of the file to be loaded.

**Warning:**

This file is located on the machine specified by the URI in the **Host** field so it should be on the same machine as the database server.

**Advanced settings**

| **Action** | Select the action to be carried out

**Bulk insert**  **Bulk update**  **Bcp query out** Depending on the action selected, the required information varies. |
| **Fire Triggers** | Select this check box to execute any insert trigger defined on the table into which data will be loaded during the bulk insert operation. This property is available when **Bulk insert** is selected from the **Action** drop-down list. |
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Additional JDBC parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulk insert &amp; Bulk update</td>
<td>Fields terminated</td>
</tr>
<tr>
<td></td>
<td>Rows terminated</td>
</tr>
<tr>
<td></td>
<td>First row</td>
</tr>
<tr>
<td></td>
<td>Code page</td>
</tr>
<tr>
<td></td>
<td>Data file type</td>
</tr>
<tr>
<td></td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td>tStatCatcher Statistics</td>
</tr>
<tr>
<td>Bcp query out</td>
<td>Fields terminated</td>
</tr>
<tr>
<td></td>
<td>Rows terminated</td>
</tr>
<tr>
<td></td>
<td>Data file type</td>
</tr>
<tr>
<td></td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td>tStatCatcher Statistics</td>
</tr>
</tbody>
</table>

**Dynamic settings**

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.
**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

---

### Related scenarios

For use cases in relation with **tMSSqlBulkExec**, see the following scenarios:

- **Scenario: Inserting transformed data in MySQL database** on page 2537.
- **Scenario: Inserting data in bulk in MySQL database** on page 2544.
# tMSSqlClose

Closes a transaction in the MSSql databases.

## tMSSqlClose Standard properties

These properties are used to configure tMSSqlClose running in the Standard Job framework.

The Standard tMSSqlClose component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

## Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tMssqlConnection</strong> component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

## Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with tMssql components, especially with <strong>tMssqlConnection</strong> and <strong>tMssqlCommit</strong>.</th>
</tr>
</thead>
</table>

## Dynamic settings

Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see [Scenario: Reading data from databases through context-](#).
Related scenarios

No scenario is available for the Standard version of this component yet.
tMSSqlColumnList

Lists all column names of a given MSSql table.
tMSSqlColumnList iterates on all columns of a given table through a defined MS SQL connection.

**tMSSqlColumnList Standard properties**

These properties are used to configure tMSSqlColumnList running in the Standard Job framework.
The Standard tMSSqlColumnList component belongs to the Databases family.
The component in this framework is generally available.

![Note:](image)

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tMSSqlConnection component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Table name</td>
<td>Enter the name of the table.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

**Global Variables**

**Global Variables**

<table>
<thead>
<tr>
<th>COLUMN_NAME: the name of the column currently iterated upon. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMN_DEFAULT: the default value of the column currently iterated upon. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>IS_NULLABLE: nullability of the column currently iterated upon. This is a Flow variable and it returns YES if the column allows NULL; otherwise, NO.</td>
</tr>
<tr>
<td>DATA_TYPE: the data type of the column currently iterated upon. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>CHARACTER_MAXIMUM_LENGTH: the maximum length in number of characters of the column currently iterated</td>
</tr>
</tbody>
</table>
upon. This is a Flow variable, and it returns a numeric string for binary or character data types and NULL for other types.

**CHARACTER_OCTET_LENGTH**: the maximum length in bytes of the column currently iterated upon. This is a Flow variable, and it returns a numeric string for binary or character data types and NULL for other types.

**NUMERIC_PRECISION**: the data precision of the column currently iterated upon. This is a Flow variable, and it returns a numeric string for approximate numeric data, exact numeric data, integer data, or monetary data and NULL for other data types.

**NUMERIC_PRECISION_RADIX**: the data precision radix of the column currently iterated upon. This is a Flow variable, and it returns a numeric string for approximate numeric data, exact numeric data, integer data, or monetary data and NULL for other data types.

**NUMERIC_SCALE**: the scale in digits allowed in the column currently iterated upon. This is a Flow variable, and it returns a numeric string for approximate numeric data, exact numeric data, integer data, or monetary data and NULL for other data types.

**DATETIME_PRECISION**: the datetime precision in fractional seconds of the column currently iterated upon. This is a Flow variable, and it returns a numeric string if the data type is datetime or smalldatetime; otherwise, it returns NULL.

**COLUMN_KEY**: the key indicator of the column currently iterated upon. This is a Flow variable and it returns a string if the column is set to Key; otherwise it returns nothing.

**NB_COLUMN**: the number of columns iterated upon so far. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | This component is to be used along with MSSql components, especially with tMSSqlConnection. |
## Dynamic settings

Click the [+](https://example.com) button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Related scenario

For **tMSSqlColumnList** related scenario, see Scenario: Iterating on a DB table and listing its column names on page 2470.
**tMSSqlCommit**

Commits in one go, using a unique connection, a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

*tMSSqlCommit* validates the data processed through the job into the connected DB.

**tMSSqlCommit Standard properties**

These properties are used to configure *tMSSqlCommit* running in the Standard Job framework. The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <em>tMSSqlConnection</em> component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task. <strong>Warning:</strong> If you want to use a <em>Row &gt; Main</em> connection to link <em>tMSSqlCommit</em> to your Job, your data will be committed row by row. In this case, do not select the <em>Close connection</em> check box or your connection will be closed before the end of your first row commit.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the job processing metadata at a job level as well as at each component level. |
### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component is more commonly used with other tMSSql* components, especially with the tMSSqlConnection and tMSSqlRollback components.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dynamic settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>

### Related scenarios

For a tMSSqlCommit related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476.
tMSSqlConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

tMSSqlConnection opens a connection to a Microsoft SQL Server database or a Microsoft Azure SQL database.

**tMSSqlConnection Standard properties**

These properties are used to configure tMSSqlConnection running in the Standard Job framework.

The Standard tMSSqlConnection component belongs to the Databases and the ELT families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>JDBC Provider</strong></td>
<td>Select the provider of the JDBC driver to be used, either <strong>Microsoft</strong> (recommended) or <strong>Open source JTDS</strong>.</td>
</tr>
<tr>
<td></td>
<td>When using this component with Datasource in Talend Runtime, you need to use the <strong>Open source JTDS</strong> driver.</td>
</tr>
<tr>
<td></td>
<td>Note that when <strong>Microsoft</strong> is selected, you need to download the Microsoft JDBC Driver for SQL Server on Microsoft Download Center, unpack the downloaded zip file, choose a jar in the unzipped folder based on your JRE version, rename the jar to mssql-jdbc.jar and install it manually. For more information about choosing the jar, see the System Requirements information on Microsoft Download Center.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Schema name.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Additional JDBC parameters</strong></td>
<td>Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair. For example, <code>encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;</code> for Azure SQL database connection.</td>
</tr>
<tr>
<td><strong>Use or register a shared DB Connection</strong></td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the <strong>Shared DB Connection Name</strong> field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child. This option is incompatible with the <strong>Use dynamic job</strong> and <strong>Use an independent process to run sub job</strong> options of the tRunJob component. Using a shared connection together with a tRunJob component with either of these two options enabled will cause your Job to fail. This check box is not available when the <strong>Specify a data source alias</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Specify a data source alias</strong></td>
<td>Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503. This check box is not visible when the <strong>Use or register a shared DB Connection</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Data source alias</strong></td>
<td>Enter the alias of the data source created on the Talend Runtime side. This field is available only when the <strong>Specify a data source alias</strong> check box is selected.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **Auto Commit** | Select this check box to commit any changes to the database automatically upon the transaction. With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. |
By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

**tStatCatcher Statistics**

Select this check box to gather the job processing metadata at a Job level as well as at each component level.

**Usage**

**Usage rule**

This component is more commonly used with other tMSSql* components, especially with the tMSSqlCommit and tMSSqlRollback components.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

**Scenario: Inserting data into a database table and extracting useful information from it**

The scenario describes a Job that reads the employee data from a text file, inserts the data into a table of an MSSQL database, then extracts useful data from the table, and displays the information on the console.
This scenario involves the following components:

- **tMSSqlConnection**: establishes a connection to the MSSQL server.
- **tFileInputDelimited**: reads the input file, defines the data structure and sends it to the next component.
- **tMSSqlOutput**: writes data it receives from the preceding component into a table of an MSSQL database.
- **tMSSqlInput**: extracts data from the table based on an SQL query.
- **tLogRow**: displays the information it receives from the preceding component on the console.
- **tMSSqlCommit**: commits the transaction in the connected MSSQL server.

**Setting up the Job**

**Procedure**

1. Drop the following components from the Palette onto the design workspace: **tMSSqlConnection**, **tFileInputDelimited**, **tMSSqlOutput**, **tMSSqlInput**, **tLogRow**, and **tMSSqlCommit**.
2. Connect **tMSSqlConnection** to **tFileInputDelimited** using a Trigger > OnSubjobOk link.
3. Do the same to connect **tFileInputDelimited** to **tMSSqlInput** and **tMSSqlCommit**.
4. Connect **tFileInputDelimited** to **tMSSqlOutput** using a Row > Main link.
5. Do the same to connect **tMSSqlInput** to **tLogRow**.
Configuring the components

Opening a connection to the MSSQL server

Procedure

1. Double-click the tMSSqlConnection component to open its Basic settings view in the Component tab.

   ![tMSSqlConnection](image)

   - **Host** field: Type in the IP address or hostname of the MSSQL server, 192.168.30.47 in this example.
   - **Port** field: Type in the port number of the database server, 1433 in this example.
   - **Schema** field: Type in the schema name, dbo in this example.
   - **Database** field: Type in the database name, talend in this example.
   - **Username** and **Password** fields: Enter the credentials for the MSSQL connection.

2. In the **Host** field, type in the IP address or hostname of the MSSQL server, 192.168.30.47 in this example.
3. In the **Port** field, type in the port number of the database server, 1433 in this example.
4. In the **Schema** field, type in the schema name, dbo in this example.
5. In the **Database** field, type in the database name, talend in this example.
6. In the **Username** and **Password** fields, enter the credentials for the MSSQL connection.

Reading the input data

Procedure

1. Double-click the tFileInputDelimited component to open its Component view.

   ![tFileInputDelimited](image)

   - **File Name/Stream** field: Click the [...] button next to the File Name/Stream field to browse to the input file. In this example, it is D:/Input/Employee_Wage.txt. This text file holds three columns: id, name and wage. id;name;wage 51;Harry;2300 40;Ronald;3796 17;Theodore;2174 21;James;1986
3. In the **Header** field, type in 1 to skip the first row of the input file.

4. Click **Edit schema** to define the data to pass on to the `tMSSqlOutput` component. In this example, we define `id` as the key, and specify the length and precision for each column respectively.

Click **OK** to close the schema editor. A dialog box opens, and you can choose to propagate the schema to the next component.

Related topic: `tFileInputDelimited` on page 1010.

**Writing the data into the database table**

**Procedure**

1. Double-click the `tMSSqlOutput` component to open its **Basic settings** view in the **Component** tab.

2. Type in required information for the connection or use the existing connection you have configured before. In this example, we select the **Use an existing connection** check box. If multiple connections are available, select the connection you want to use from the **Component List** drop-down list.

3. In the **Table** field, type in the name of the table you want to write the data to: `Wage_Info` in this example. You can also click the `[...]` button next to the **Table** field to open a dialog box and select a proper table.

4. Select **Create table if not exists** from the **Action on table** drop-down list.

5. Select **Insert if not exists** from the **Action on data** drop-down list.

6. Click **Sync columns** to retrieve the schema from the preceding component.
Extracting useful information from the table

Procedure

1. Double-click the **tMSSqlInput** component to open its **Basic settings** view in the **Component** tab.

![tMSSqlInput component with Basic settings](image)

2. Select the **Use an existing connection** check box. If multiple connections are available, select the connection you want to use from the **Component List** drop-down list.

3. Click **Edit schema** to define the data structure to be read from the table. In this example, we need to read all three columns from the table.

![Schema of tMSSqlInput component](image)

4. In the **Table Name** field, type in the name of the table you want to read the data from: *Wage_Info* in this example.

5. In the **Query** field, fill in the SQL query to be executed on the table specified. To obtain the data of employees whose wages are above the average value and order them by id, enter the SQL query as

```sql
SELECT * 
FROM Wage_Info 
WHERE wage > (SELECT avg(wage) FROM Wage_Info) 
ORDER BY id
```
follows:

```
SELECT * FROM Wage_Info WHERE wage > (SELECT avg(wage) FROM Wage_Info) ORDER BY id
```

**Displaying information on the console**

**Procedure**

1. Double-click the `tLogRow` component to open its **Basic settings** view.
2. In the **Mode** area, select **Table (print values in cells of a table)**.

**Committing the transaction and closing the connection**

**Procedure**

1. Double-click the `tMSSqlCommit` component to open its **Basic settings** view.
   - 2. Select the **Close Connection** check box.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Execute the Job by pressing **F6** or clicking **Run** on the **Run** tab.

The information of employees whose wages are above the average value ordered by id is displayed on the console.
tMSSqlInput

Executes a DB query with a strictly defined order which must correspond to the schema definition.

tMSSqlInput reads data and extracts fields based on a query from a Microsoft SQL Server database or a Microsoft Azure SQL database. Then it passes on the field list to the next component via a Main row link.

**tMSSqlInput Standard properties**

These properties are used to configure tMSSqlInput running in the Standard Job framework.

The Standard tMSSqlInput component belongs to the Databases family.

The component in this framework is generally available.

<i>Note:</i>

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>JDBC Provider</strong></td>
<td>Select the provider of the JDBC driver to be used, either Microsoft (recommended) or Open source JTDS.</td>
</tr>
<tr>
<td></td>
<td>When using this component with Datasource in Talend Runtime, you need to use the Open source JTDS driver.</td>
</tr>
<tr>
<td></td>
<td>Note that when Microsoft is selected, you need to download the Microsoft JDBC Driver for SQL Server on Microsoft Download Center, unpack the downloaded zip file, choose a jar in the unzipped folder based on your JRE version, rename the jar to mssql-jdbc.jar and install it manually. For more information about choosing the jar,</td>
</tr>
</tbody>
</table>
Use an existing connection

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

Note:

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>Name of the schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
</tbody>
</table>

- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available.
- **View schema**: choose this option to view the schema only.

- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Query type and Query**

Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

**Specify a data source alias**

Select this check box and specify the alias of a data source created on the *Talend Runtime* side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in *Talend Runtime*. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

This check box is not available when the **Use an existing connection** check box is selected.

**Data source alias**

Enter the alias of the data source created on the *Talend Runtime* side.

This field is available only when the **Specify a data source alias** check box is selected.

**Advanced settings**

**Additional JDBC parameters**

Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair. For example, `encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;` for Azure SQL database connection.

This field is not available if the **Use an existing connection** check box is selected.

**Trim all the String/Char columns**

Select this check box to remove leading and trailing whitespace from all the String/Char columns.

**Trim column**

Remove leading and trailing whitespace from defined columns.

**tStat Catcher Statistics**

Select this check box to collect log data at the component level.
### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>QUERY: the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible SQL queries for MS SQL server databases.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Dynamic settings</th>
<th>Click the <strong>[+]</strong> button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The <strong>Dynamic settings</strong> table is available only when the <strong>Use an existing connection</strong> check box is selected in the <strong>Basic settings</strong> view. Once a dynamic parameter is defined, the <strong>Component List</strong> box in the <strong>Basic settings</strong> view becomes unusable.</td>
</tr>
<tr>
<td></td>
<td>For examples on using dynamic parameters, see <em>Scenario: Reading data from databases through context-based dynamic connections</em> on page 2497 and <em>Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters</em> on page 489. For more information on <strong>Dynamic settings</strong> and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find |
Related scenarios

For related topics, see:

- tMSSqlConnection on page 2406

For related topic in tContextLoad, see Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
**tMSSqlLastInsertId**

Retrieves the last primary keys added by a user to a MSSql table.

`tMSSqlLastInsertId` displays the last IDs added to a table from a MSSql specified connection.

### tMSSqlLastInsertId Standard properties

These properties are used to configure `tMSSqlLastInsertId` running in the Standard Job framework.

The Standard `tMSSqlLastInsertId` component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema</strong> and <strong>Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In:</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
<tr>
<td><strong>Repository:</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.
<table>
<thead>
<tr>
<th>Component list</th>
<th>Select the <strong>tMSSqlConnection</strong> component on the <strong>Component list</strong> to reuse the connection details you already defined, if there are more than one component in this list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced settings</td>
<td></td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
<tr>
<td>Global Variables</td>
<td></td>
</tr>
</tbody>
</table>
| Global Variables | **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
*For further information about variables, see [Talend Studio User Guide](#).* |
| Usage | | |
| Usage rule | This component offers the flexibility of the DB query and covers all possible SQL queries. |
| Dynamic settings | Click the [*] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  
Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.  
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.  
For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |

2420
<table>
<thead>
<tr>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the <strong>Install</strong> button on the <strong>Component</strong> tab view. You can also find out and add all missing JARs easily on the <strong>Modules</strong> tab in the <strong>Integration</strong> perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

**Related scenario**

For a related scenario, see *Scenario: Get the ID for the last inserted record with tMysqlLastInsertId* on page 2510
tMSSqlOutput

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tMSSqlOutput writes, updates, makes changes or suppresses entries in a Microsoft SQL Server database or a Microsoft Azure SQL database.

**tMSSqlOutput Standard properties**

These properties are used to configure tMSSqlOutput running in the Standard Job framework.

The Standard tMSSqlOutput component belongs to the Databases family.

The component in this framework is generally available.

Note: This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td></td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.</td>
</tr>
<tr>
<td></td>
<td>For more information about setting up and storing database connection parameters, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. |
| | Note: When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to: |
| | 1. In the parent level, register the database connection to be shared in the Basic |
settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

| JDBC Provider | Select the provider of the JDBC driver to be used, either **Microsoft** (recommended) or **Open source JTDS**.  
When using this component with Datasource in Talend Runtime, you need to use the **Open source JTDS** driver.  
Note that when **Microsoft** is selected, you need to download the Microsoft JDBC Driver for SQL Server on [Microsoft Download Center](https://www.microsoft.com/download), unpack the downloaded zip file, choose a jar in the unzipped folder based on your JRE version, rename the jar to `mssql-jdbc.jar` and install it manually. For more information about choosing the jar, see the System Requirements information on [Microsoft Download Center](https://www.microsoft.com/download). |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Schema</td>
<td>Name of the schema.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
</tbody>
</table>
| Username and Password | DB user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| Table | Name of the table to be written. Note that only one table can be written at a time. |
| Action on table | On the table defined, you can perform one of the following operations:  
**Default**: No operation is carried out.  
**Drop and create table**: The table is removed and created again.  
**Create table**: The table does not exist and gets created.  
**Create table if not exists**: The table is created if it does not exist.  
**Drop table if exists and create**: The table is removed if it already exists and created again.  
**Clear table**: The table content is deleted.  
**Truncate table**: The table content is deleted. You do not have the possibility to rollback the operation. |
<table>
<thead>
<tr>
<th>Turn on identity insert</th>
<th>Select this check box to use your own sequence for the identity value of the inserted records (instead of having the SQL Server pick the next sequential value).</th>
</tr>
</thead>
</table>
| Action on data | On the data of the table defined, you can perform:  
- **Insert**: Add new entries to the table. If duplicates are found, job stops.  
- **Single Insert Query**: Add entries to the table in a batch  
- **Update**: Make changes to existing entries  
- **Insert or update**: Insert a new record. If the record with the given reference already exists, an update would be made.  
- **Update or insert**: Update the record with the given reference. If the record does not exist, a new record would be inserted.  
- **Delete**: Remove entries corresponding to the input flow.  
- **Insert if not exist**: Add new entries to the table if they do not exist.  

**Warning:**  
It is necessary to specify at least one column as a primary key on which the **Update** and **Delete** operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the **Update** and **Delete** operations. To do that: Select the **Use field options** check box and then in the **Key in update column**, select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the **Key in delete column** for the Delete operation. |
| Specify identity field | Select this check box to specify the identity field, which is made up of an automatically incrementing identification number.  
When this check box is selected, three other fields are displayed:  
- **Identity field**: select the column you want to define as the identity field from the list.  
- **Start value**: type in a start value, used for the very first row loaded into the table.  
- **Step**: type in an incremental value, added to the value of the previous row that was loaded.  

This check box is available only when you select **Drop and create table**, **Create table**, **Create table if not exists**, or **Drop table if exists and create** from the **Action on table** list, and will disappear if you select the **Enable parallel execution** check box in the **Advanced settings** view. If you select this check box with the **Turn on identity insert** check box cleared and the **Create table if
The `not exists` option selected from the **Action on table** list and if the specified table does not exist, only a table will be created without inserting data into it.

### Note:
You can also specify the identity field from the schema of the component. To do so, set the **DB Type** of the relevant column to **INT IDENTITY**.

### Note:
When the **Specify identity field** check box is selected, the **INT IDENTITY DB Type** in the schema is ignored.

#### Schema and Edit schema
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word *line* when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

#### Built-In
You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](https://help.talend.com).

#### Repository
You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](https://help.talend.com).

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in [Talend Help Center](https://help.talend.com).

#### Specify a data source alias
Select this check box and specify the alias of a data source created on the **Talend Runtime** side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in **Talend Runtime**. For a related use case, see [Scenario: Deploying your Job on Talend](https://help.talend.com).
### Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data source alias</strong></td>
<td>Enter the alias of the data source created on the Talend Runtime side. This field is available only when the Specify a data source alias check box is selected.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row &gt; Rejects link.</td>
</tr>
<tr>
<td><strong>Additional JDBC parameters</strong></td>
<td>Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair. For example, <code>encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;</code> for Azure SQL database connection. This field is not available if the Use an existing connection check box is selected.</td>
</tr>
<tr>
<td><strong>Commit every</strong></td>
<td>Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.</td>
</tr>
<tr>
<td><strong>Additional Columns</strong></td>
<td>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</td>
</tr>
<tr>
<td>Name</td>
<td>Type in the name of the schema column to be altered or inserted as new column.</td>
</tr>
<tr>
<td>SQL expression</td>
<td>Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td>Position</td>
<td>Select Before, Replace or After following the action to be performed on the reference column.</td>
</tr>
<tr>
<td>Reference column</td>
<td>Type in a column of reference that the tDBOutput can use to place or replace the new or altered column.</td>
</tr>
<tr>
<td>Use field options</td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td>Ignore date validation</td>
<td>Select this check box to ignore the date validation and insert the data directly into the database for the</td>
</tr>
<tr>
<td><strong>Global Variables</strong></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td><strong>Global Variables</strong></td>
<td><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_UPDATED</strong>: the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_DELETED</strong>: the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_REJECTED</strong>: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>QUERY</strong>: the query statement processed. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions</td>
</tr>
</tbody>
</table>
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide.*

---

**Usage**

**Usage rule**

This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a MSSql database. It also allows you to create a reject flow using a **Row > Rejects** link to filter data in error. For an example of the use of `tMysqlOutput` in use, see *Scenario: Retrieving data in error with a Reject link* on page 2529.

**Dynamic settings**

Click the `[*]` button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide.*

**Limitation**

When the **Single Insert Query** option is selected in the **Action on data** list, an SQL Prepared Statement is generated, for example, `INSERT INTO table (col1, col2, col3) VALUES (?,?,?) , (?,?,?) , (?,?,?) , (?,?,?)`. Within brackets are the groups of parameters the number of which cannot exceed 2000, generally, depending on the JDBC driver. Therefore, the batch size should be set so that this limit is respected.
Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

**Related scenarios**

For **tMSSqlOutput** related topics, see:

- **tMSSqlConnection** on page 2406.
- **Scenario: Inserting a column and altering data using tMysqlOutput** on page 2521.
**tMSSqlOutputBulk**

 Prepares the file to be used as parameter in the INSERT query to feed the MSSql database.

 The tMSSqlOutputBulk and tMSSqlBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tMSSqlOutputBulkExec component, detailed in a separate section. The advantage of using a two step process is that the data can be transformed before it is loaded in the database.

 tMSSqlOutputBulk writes a file with columns based on the defined delimiter and the MSSql standards.

**tMSSqlOutputBulk Standard properties**

 These properties are used to configure tMSSqlOutputBulk running in the Standard Job framework. The Standard tMSSqlOutputBulk component belongs to the Databases family.

 The component in this framework is generally available.

 _Note:_

 This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

 For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>File Name</td>
<td>Name of the file to be generated.</td>
</tr>
<tr>
<td><strong>Warning</strong>:</td>
<td></td>
</tr>
<tr>
<td>This file is generated on the local machine or a shared folder on the LAN.</td>
<td></td>
</tr>
<tr>
<td>Append</td>
<td>Select this check box to add the new rows at the end of the records.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
</tbody>
</table>
Built-In: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Advanced settings

<table>
<thead>
<tr>
<th>Row separator</th>
<th>String (ex: &quot;\n&quot; on Unix) to distinguish rows.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field separator</td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td>Include header</td>
<td>Select this check to include the column header.</td>
</tr>
<tr>
<td>Encoding</td>
<td>Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td>tStaCatcher statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

**Global Variables**

- **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with <em>tMSSqlBulkExec</em> component. Used together they offer gains in performance while feeding a MSSql database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component family</td>
<td>Databases/MS SQL Server</td>
</tr>
</tbody>
</table>

## Related scenarios

For use cases in relation with *tMSSqlOutputBulk*, see the following scenarios:

- **Scenario: Inserting transformed data in MySQL database** on page 2537.
- **Scenario: Inserting data in bulk in MySQL database** on page 2544.
tMSSqlOutputBulkExec

Gains in performance during Insert operations to a Microsoft SQL Server database.

The tMSSqlOutputBulk and tMSSqlBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tMSSqlOutputBulkExec component.

**tMSSqlOutputBulkExec Standard properties**

These properties are used to configure tMSSqlOutputBulkExec running in the Standard Job framework. The Standard tMSSqlOutputBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components on page 598](#).

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Select the action to be carried out</td>
</tr>
<tr>
<td></td>
<td>Bulk insert  Bulk update</td>
</tr>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository .</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

| **JDBC Provider** | Select the provider of the JDBC driver to be used, either **Microsoft** (recommended) or **Open source JTDS**. When using this component with Datasource in Talend Runtime, you need to use the **Open source JTDS** driver. Note that when **Microsoft** is selected, you need to download the Microsoft JDBC Driver for SQL Server on Microsoft Download Center, unpack the downloaded zip file, choose a jar in the unzipped folder based on your JRE version, rename the jar to *mssql-jdbc.jar* and install it manually. For more information about choosing the jar, see the System Requirements information on Microsoft Download Center. |
| **Host** | Database server IP address. Currently, only *localhost, 127.0.0.1* or the exact IP address of the local machine is allowed for proper functioning. In other words, the database server must be installed on the same machine where the Studio is installed or where the Job using *tMSSqlOutputBulkExec* is deployed. |
| **Port** | Listening port number of DB server. |
| **DB name** | Name of the database |
| **Schema** | Name of the schema. |
| **Username** and **Password** | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Table** | Name of the table to be written. Note that only one table can be written at a time and that the table must exist for the insert operation to succeed. |
| **Action on table** | On the table defined, you can perform one of the following operations: **None**: No operation is carried out. **Drop and create a table**: The table is removed and created again. **Create a table**: The table does not exist and gets created. **Create a table if not exists**: The table is created if it does not exist. |
### Truncate table
The table content is deleted. You do not have the possibility to rollback the operation.

### Clear a table
The table content is deleted. You have the possibility to rollback the operation.

### Schema and Edit schema
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

#### Built-In
You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](https://help.talend.com).

#### Repository
You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](https://help.talend.com).

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content](https://help.talend.com) window.

### File Name
Name of the file to be generated and loaded.

**Warning:**
This file is generated on the machine specified by the URI in the **Host** field so it should be on the same machine as the database server.

### Append
Select this check box to add the new rows at the end of the records.

### Advanced settings

#### Additional JDBC parameters
Specify additional connection properties for the database connection you are creating. The properties
are separated by semicolon and each property is a key-value pair. For example, `encrypt=true;trustServerCertificate=false; hostNameInCertificate=*.database.windows.net;loginTimeout=30;` for Azure SQL database connection.

This field is not available if the **Use an existing connection** check box is selected.

<table>
<thead>
<tr>
<th>Field separator</th>
<th>Character, string or regular expression to separate fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row separator</td>
<td>String (ex: <code>\n</code> on Unix) to distinguish rows.</td>
</tr>
<tr>
<td>First row</td>
<td>Type in the number of the row where the action should start.</td>
</tr>
<tr>
<td>Include header</td>
<td>Select this check box to include the column header.</td>
</tr>
<tr>
<td>Code page</td>
<td><strong>OEM</strong> code pages used to map a specific set of characters to numerical code point values.</td>
</tr>
<tr>
<td>Data file type</td>
<td>Select the type of data being handled.</td>
</tr>
<tr>
<td>Encoding</td>
<td>Select the encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td><strong>tStatCatcher statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Usage

#### Usage rule

This component is mainly used when no particular transformation is required on the data to be loaded onto the database.

#### Dynamic settings

Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489.

For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.
**Limitation**

The database server must be installed on the same machine where the Studio is installed or where the Job using `tMSSqlOutputBulkExec` is deployed, so that the component functions properly.

---

**Related scenarios**

For use cases in relation with `tMSSqlOutputBulkExec`, see the following scenarios:

- **Scenario: Inserting transformed data in MySQL database** on page 2537
- **Scenario: Inserting data in bulk in MySQL database** on page 2544
tMSSqlRollback

Cancels the transaction commit in the MSSql database and thus avoids to commit part of a transaction involuntarily.

**tMSSqlRollback Standard properties**

These properties are used to configure tMSSqlRollback running in the Standard Job framework. The Standard tMSSqlRollback component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tMSSqlConnection component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tMSSql components, especially with the tMSSqlConnection and tMSSqlCommit components. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |
Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenario

For tMSSqlRollback related scenario, see Scenario: Rollback from inserting data in mother/daughter tables on page 2480.
tMSSqlRow

Acts on the actual DB structure or on the data (although without handling data).

`tMSSqlRow` executes the SQL query stated onto the specified Microsoft SQL Server or Azure SQL database, depending on the nature of the query and the database. The row suffix means the component implements a flow in the job design although it doesn’t provide output.

The `SQLBuilder` tool helps you write easily your SQL statements.

**tMSSqlRow Standard properties**

These properties are used to configure `tMSSqlRow` running in the Standard Job framework.

The Standard `tMSSqlRow` component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

**JDBC Provider**
Select the provider of the JDBC driver to be used, either Microsoft (recommended) or Open source JTDS.
When using this component with Datasource in Talend Runtime, you need to use the Open source JTDS driver.
Note that when Microsoft is selected, you need to download the Microsoft JDBC Driver for SQL Server on Microsoft Download Center, unpack the downloaded zip file, choose a jar in the unzipped folder based on your JRE version, rename the jar to mssql-jdbc.jar and install it manually. For more information about choosing the jar, see the System Requirements information on Microsoft Download Center.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
<td></td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In:</strong> You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository:</strong> You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
<td></td>
</tr>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate</td>
<td></td>
</tr>
</tbody>
</table>
the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Table name</th>
<th>Name of the table to be used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn on identity insert</td>
<td>Select this check box to use your own sequence for the identity value of the inserted records (instead of having the SQL Server pick the next sequential value).</td>
</tr>
<tr>
<td>Query type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in</td>
<td>Fill in manually the query statement or build it graphically using SQLBuilder.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</td>
</tr>
<tr>
<td>Guess Query</td>
<td>Click the <strong>Guess Query</strong> button to generate the query which corresponds to your table schema in the Query field.</td>
</tr>
<tr>
<td>Query</td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td>Specify a data source alias</td>
<td>Select this check box and specify the alias of a data source created on the <strong>Talend Runtime</strong> side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in <strong>Talend Runtime</strong>. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503. This check box is not available when the <strong>Use an existing connection</strong> check box is selected.</td>
</tr>
<tr>
<td>Data source alias</td>
<td>Enter the alias of the data source created on the <strong>Talend Runtime</strong> side. This field is available only when the <strong>Specify a data source alias</strong> check box is selected.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| Additional JDBC parameters | Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair. For example, `encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30` for Azure SQL database connection. |
This field is not available if the **Use an existing connection** check box is selected.

| Propagate QUERY's recordset | Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the **use column** list.  
| Note: | This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY's recordset should be set to the type of **Object** and this component is usually followed by **tParseRecordSet**. |

| Use PreparedStatement | Select this check box if you want to query the database using a PreparedStatement. In the **Set PreparedStatement Parameter** table, define the parameters represented by "?" in the SQL instruction of the **Query** field in the **Basic Settings** tab.  
| Parameter Index: | Enter the parameter position in the SQL instruction.  
| Parameter Type: | Enter the parameter type.  
| Parameter Value: | Enter the parameter value.  
| Note: | This option is very useful if you need to execute the same query several times. Performance levels are increased |

| Commit every | Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions. |

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

### Global Variables

| Global Variables | **QUERY**: the query statement being processed. This is a Flow variable and it returns a string.  
| **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
| A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
| To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. |

| Global Variables | **QUERY**: the query statement being processed. This is a Flow variable and it returns a string.  
| **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
| A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
| To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. |
Usage

Usage rule
This component offers the flexibility of the DB query and covers all possible SQL queries.

Dynamic settings
Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Limitation
Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Related scenarios

For related topics, see:

- Scenario: Combining two flows for selective output on page 2558
- Procedure on page 625.
- Scenario: Removing and regenerating a MySQL table index on page 2552.
tMSSqlSCD

Tracks and reflects changes in a dedicated SCD table in a Microsoft SQL Server or Azure SQL database. tMSSqlSCD addresses Slowly Changing Dimension needs, reading regularly a source of data and logging the changes into a dedicated SCD table.

**tMSSqlSCD Standard properties**

These properties are used to configure tMSSqlSCD running in the Standard Job framework. The Standard tMSSqlSCD component belongs to the Business Intelligence and the Databases families. The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the Repository file where Properties are stored. The following fields are pre-filled in using fetched data.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
</tbody>
</table>
For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](https://docs.talend.com).

<table>
<thead>
<tr>
<th>JDBC Provider</th>
<th>Select the provider of the JDBC driver to be used, either <strong>Microsoft</strong> (recommended) or <strong>Open source JTDS</strong>. When using this component with Datasource in Talend Runtime, you need to use the <strong>Open source JTDS</strong> driver. Note that when <strong>Microsoft</strong> is selected, you need to download the Microsoft JDBC Driver for SQL Server on <a href="https://www.microsoft.com">Microsoft Download Center</a>, unpack the downloaded zip file, choose a jar in the unzipped folder based on your JRE version, rename the jar to <code>mssql-jdbc.jar</code> and install it manually. For more information about choosing the jar, see the System Requirements information on <a href="https://www.microsoft.com">Microsoft Download Center</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server</strong></td>
<td><strong>Database server IP address.</strong></td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td><strong>Listening port number of DB server.</strong></td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td><strong>Name of the DB schema.</strong></td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td><strong>Name of the database.</strong></td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td><strong>DB user authentication data.</strong> To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td><strong>Name of the table to be written. Note that only one table can be written at a time.</strong></td>
</tr>
</tbody>
</table>
| **Schema and Edit schema** | **A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the **Repository** type, three options are available:**
| | • **View schema**: choose this option to view the schema only.
| | • **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
| | • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window. |
| **Built-in:** | The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*. |
| **Repository:** | The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*. |

**SCD Editor**
The SCD editor helps to build and configure the data flow for slowly changing dimension outputs.
For more information, see *SCD management methodology* on page 2566.

**Use memory saving Mode**
Select this check box to maximize system performance.

**Source keys include Null**
Select this check box to allow the source key columns to have Null values.

**Warning:**
Special attention should be paid to the uniqueness of the source key(s) values when this option is selected.

**Die on error**
This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

**Advanced settings**

**Additional JDBC parameters**
Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair. For example, `encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;` for Azure SQL database connection.
This field is not available if the **Use an existing connection** check box is selected.

**End date time details**
Specify the time value of the SCD end date time setting in the format of `HH:mm:ss`. The default value for this field is `12:00:00`.
This field appears only when SCD Type 2 is used and **Fixed year value** is selected for creating the SCD end date.

**tStatCatcher Statistics**
Select this check box to collect log data at the component level.

**Debug mode**
Select this check box to display each step during processing entries in a database.

**Global Variables**

| **Global Variables** | **NB_LINE_UPDATED**: the number of rows updated. This is an After variable and it returns an integer. |
**NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.

**NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

---

**Usage**

**Usage rule**

This component is used as Output component. It requires an Input component and Row main link as input.

**Dynamic settings**

Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections on page 2497* and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489*. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Limitation**

This component does not support using SCD type 0 together with other SCD types.

---

**Related scenario**

For related topics, see *tMysqlSCD on page 2563*. 
tMSSqlSP

Offers a convenient way to centralize multiple or complex queries in a database and calls them easily. tMSSqlSP calls the stored procedure in a Microsoft SQL Server or Azure SQL database.

**tMSSqlSP Standard properties**

These properties are used to configure tMSSqlSP running in the Standard Job framework.

The Standard tMSSqlSP component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>. <strong>Built-in</strong>: No property data stored centrally. <strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined. <strong>Note:</strong> When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to: 1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection. 2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
| **JDBC Provider** | Select the provider of the JDBC driver to be used, either **Microsoft** (recommended) or **Open source JTDS**.  
When using this component with Datasource in Talend Runtime, you need to use the **Open source JTDS** driver.  
Note that when **Microsoft** is selected, you need to download the Microsoft JDBC Driver for SQL Server on Microsoft Download Center, unpack the downloaded zip file, choose a jar in the unzipped folder based on your JRE version, rename the jar to *mssql-jdbc.jar* and install it manually. For more information about choosing the jar, see the System Requirements information on Microsoft Download Center. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Name of the schema.</td>
</tr>
</tbody>
</table>
| **Username and Password** | DB user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Schema and Edit Schema** | In SP principle, the schema is an input parameter.  
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. |
| **Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*. | |
| **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*. | |
| Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** |
upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>SP Name</th>
<th>Type in the exact name of the Stored Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is Function / Return result</td>
<td>Select this box, if only a value is to be returned. Select on the list the schema column, the value to be returned is based on.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Click the Plus button and select the various <strong>Schema Columns</strong> that will be required by the procedures. Note that the SP schema can hold more columns than there are parameters used in the procedure. Select the <strong>Type</strong> of parameter: <strong>IN</strong>: Input parameter <strong>OUT</strong>: Output parameter/return value <strong>IN OUT</strong>: Input parameters is to be returned as value, likely after modification through the procedure (function). <strong>RECORDSET</strong>: Input parameters is to be returned as a set of values, rather than single value.</td>
</tr>
<tr>
<td>Specify a data source alias</td>
<td>Select this box and specify the alias of a data source created on the <strong>Talend Runtime</strong> side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in <strong>Talend Runtime</strong>. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503. This check box is not available when the <strong>Use an existing connection</strong> check box is selected.</td>
</tr>
<tr>
<td>Data source alias</td>
<td>Enter the alias of the data source created on the <strong>Talend Runtime</strong> side. This field is available only when the <strong>Specify a data source alias</strong> check box is selected.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| Additional JDBC parameters | Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair. For example, `encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;` for Azure SQL database connection. |
| tMSSqlSP | This field is not available if the **Use an existing connection** check box is selected. |
| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

### Usage rule

This component is used as intermediary component. It can be used as start component but only input parameters are thus allowed.

### Dynamic settings

Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see [Scenario: Reading data from databases through context-based dynamic connections](#) on page 2497 and [Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters](#) on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Limitation

The Stored Procedures syntax should match the Database syntax.

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

---

**Scenario: Retrieving personal information using a stored procedure**

This scenario describes a Job that retrieves a personal information record from a specified table using a stored procedure according to the **id** value defined in the input flow.
In this scenario, the table to retrieve data from contains the personal information. To reproduce this scenario, you can write the data into the table from a CSV file like the following. For how to write data into a MS SQL table, see Scenario: Inserting data into a database table and extracting useful information from it on page 2408.

id;name;sex;age
1;Ford;Male;25
2;Rose;Female;30
3;Sabrina;Female;28
4;Teddy;Male;32
5;Kate;Male;35

In this scenario, the stored procedure used to retrieve the personal information is as follows:

```
CREATE PROCEDURE [dbo].[QueryPerson]
@id int,
@name varchar(50)
AS
BEGIN
SET NOCOUNT ON
SELECT * FROM dbo.person where id=@id
END
```

### Dropping and linking the components

**Procedure**

1. Create a new Job and add the following components by typing their names in the design workspace or dropping them from the Palette: a tFixedFlowInput component, a tMSSqlSP component, a tParseRecordSet component, and two tLogRow components.

2. Connect tFixedFlowInput to tMSSqlSP using a Row > Main link.

3. Do the same to connect tMSSqlSP to the first tLogRow, the first tLogRow to tParseRecordSet, and tParseRecordSet to the second tLogRow.

### Configuring the components

**Configuring the input component**

**Procedure**

1. Double-click tFixedFlowInput to open its Basic settings view.
2. Click the [...] button next to Edit schema to open the schema editor.

Click the [+] button to add two columns: id of the integer type and name of the string type. Click OK to close the schema editor.

3. In the Mode area, fill each column with respective value. In this example, the id value is set to 4, and the name value is null since it is not used in the select SQL statement in the stored procedure.

Configuring the tMSSqlSP component

Procedure

1. Double-click tMSSqlSP to open its Basic settings view.
2. Fill in the **Host**, **Port**, **Schema**, **Database**, **Username**, and **Password** fields with your MS SQL Server connection details.

3. In the **SP Name** field, enter the name of the stored procedure to be used. In this example, it is `QueryPerson`.

4. Click the `[...]` button next to **Edit schema** to open the schema editor.

Click the button to copy all columns from the input schema to the output schema, and then click the `[+]` button in the right panel to add a new column `person` of the object type to hold the personal information to be retrieved from the database.

Click **OK** to close the schema editor and accept the propagation prompted by the pop-up dialog box.
5. In the **Parameters** area, click the `[+]` button to add three rows and select a schema column and its type for each row. In this example, the *id* and *name* columns are of type IN, and the *person* column is of type RECORD SET.

**Configuring the tParseRecordSet component**

**Procedure**

1. Double-click **tParseRecordSet** to open its **Basic settings** view.

   ![Image of tParseRecordSet component](image)

   - **Prev. Comp. Column list** drop-down list, select the column that holds the personal information to be parsed. In this example, it is *person*.

   - Click the `[...]` button next to **Edit schema** to open the schema editor.

   ![Image of schema editor](image)

   - Click the `[+]` button in the right panel to add four columns: *id* and *age* of the integer type, and *name* and *sex* of the string type.

   - Click **OK** to close the schema editor and accept the propagation prompted by the pop-up dialog box.

2. In the **Attribute table** area, fill in each **Value** column with the corresponding column name in the MS SQL Server table that holds the personal information.
Configuring the output components

Procedure

1. Double-click the first tLogRow to open its Basic settings view.

   ![tLogRow_1 settings](image)

   In the Mode area, select **Vertical (each row is a key/value list)** for a better display of the result.

2. Do the same to configure the second tLogRow.

Saving and executing the Job

Procedure

1. Press **Ctrl + S** to save the Job.

2. Press **F6** to run the Job.

```
Starting job tMSSqlSP_Demo at 16:16 29/09/2014.
[statistics] connecting to socket on port 3560
[statistics] connected

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>4</td>
</tr>
<tr>
<td>name</td>
<td>null</td>
</tr>
<tr>
<td>person</td>
<td>net.sourceforge.jtde.jdbc.CachedResultSet$22e6b6b0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>4</td>
</tr>
<tr>
<td>name</td>
<td>Teddy</td>
</tr>
<tr>
<td>sex</td>
<td>Male</td>
</tr>
<tr>
<td>age</td>
<td>32</td>
</tr>
</tbody>
</table>

[statistics] disconnected
Job tMSSqlSP_Demo ended at 16:16 29/09/2014. [exit code=0]
```

The personal information with the id of 4 in the MS SQL table is displayed in the lower table on the console. Note that the values of the **id** and **name** columns shown in the upper table are the input data.
Related scenarios

For related scenarios, see:

- **Scenario: Using tMysqlSP to find a State Label using a stored procedure** on page 2583.
- **Scenario: Checking number format using a stored procedure** on page 2809.
- **Executing a stored procedure using tMDMSP** on page 2190.

Check as well **Scenario: Inserting data in mother/daughter tables** on page 2476 to analyze a set of records from a database table or DB query and return single records.
**tMSSqlTableList**

Lists the names of a given set of MSSql tables using a select statement based on a Where clause.

**tMSSqlTableList Standard properties**

These properties are used to configure tMSSqlTableList running in the Standard Job framework.

The Standard tMSSqlTableList component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tMSSqlConnection component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Where clause for table name selection</td>
<td>Enter the Where clause to identify the tables to iterate on.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>CURRENT_TABLE</strong>: the name of the table currently iterated upon. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_TABLE</strong>: the number of tables iterated upon so far. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
</tbody>
</table>
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with MSSql components, especially with tMSSqlConnection.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Related scenario

For tMSSqlTableList related scenario, see Scenario: Iterating on a DB table and listing its column names on page 2470.
tMysqlBulkExec

Offers gains in performance while executing the Insert operations on a Mysql database.

The tMysqlOutputBulk and tMysqlBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT statement used to feed a database. These two steps are fused together in the tMysqlOutputBulkExec component, detailed in a separate section. The advantage of using two separate steps is that the data can be transformed before it is loaded in the database.

tMysqlBulkExec Standard properties

These properties are used to configure tMysqlBulkExec running in the Standard Job framework.

The Standard tMysqlBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the version of Mysql that you are using.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
</tbody>
</table>

**Username and Password**

DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**Action on table**

On the table defined, you can perform one of the following operations:

- **None**: No operation is carried out.
- **Drop and create table**: The table is removed and created again.
- **Create table**: The table does not exist and gets created.
- **Create table if not exists**: The table is created if it does not exist.
- **Drop table if exists and create**: The table is removed if it already exists and created again.
- **Clear table**: The table content is deleted. You have the possibility to rollback the operation.
- **Truncate table**: The table content is deleted. You do not have the possibility to rollback the operation.

**Table**

Name of the table to be written. Note that only one table can be written at a time and that the table must exist for the insert operation to succeed.

**Local file Name**

Name of the file to be loaded. This file should be located on the same machine where the Studio is installed or where the Job using tMysqlBulkExec is deployed.

**Schema and Edit Schema**

A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional JDBC parameters</strong></td>
<td>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <em>Use an existing connection</em> check box in the Basic settings.</td>
</tr>
<tr>
<td><strong>Lines terminated by</strong></td>
<td>Character or sequence of characters used to separate lines.</td>
</tr>
<tr>
<td><strong>Fields terminated by</strong></td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td><strong>Enclosed by</strong></td>
<td>Character used to enclose text.</td>
</tr>
<tr>
<td><strong>Action on data</strong></td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert records in table</strong>: Add new records to the table.</td>
</tr>
<tr>
<td></td>
<td><strong>Update records in table</strong>: Make changes to existing records.</td>
</tr>
<tr>
<td></td>
<td><strong>Replace records in table</strong>: Replace existing records with new ones. <strong>Ignore records in table</strong>: Ignore the existing records, or insert the new ones.</td>
</tr>
<tr>
<td><strong>Records contain NULL value</strong></td>
<td>Check this box if you want to retrieve the null values from the input data flow. If you do not check this box, the null values from the input data flow will be considered as empty fields in the output data flow.</td>
</tr>
</tbody>
</table>
Encoding

Select the encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling.

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

Usage

Usage rule

This component is to be used along with tMysqlOutputBulk component. Used together, they can offer gains in performance while feeding a Mysql database.

Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Related scenarios

For use cases in relation with tMysqlBulkExec, see the following scenarios:

- Scenario: Inserting transformed data in MySQL database on page 2537.
- Scenario: Inserting data in bulk in MySQL database on page 2544.
• Scenario: Truncating and inserting file data into an Oracle database on page 2752.
tMysqlClose

Closes the transaction committed in a Mysql database.

**tMysqlClose Standard properties**

These properties are used to configure tMysqlClose running in the Standard Job framework.

The Standard tMysqlClose component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Component list</strong></td>
<td>Select the <strong>tMysqlConnection</strong> component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

### Advanced settings

| **tStat Catcher Statistics** | Select this check box to collect log data at the component level. |

### Usage

| **Usage rule** | This component is to be used along with Mysql components, especially with **tMysqlConnection** and **tMysqlCommit**. |

| **Dynamic settings** | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context- |
Related scenario

This component is closed related to tMysqlConnection and tMysqlRollback. It is often used along with a tMysqlConnection compoent since it closes an ongoing transaction.

For a scenario related to tMysqlClose, see tMysqlConnection on page 2475.
**tMysqlColumnList**

Iterates on all columns of a given Mysql table and lists column names.

**tMysqlColumnList Standard properties**

These properties are used to configure tMysqlColumnList running in the Standard Job framework.

The Standard tMysqlColumnList component belongs to the Databases family.

The component in this framework is generally available.

![Note:]

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see *Dynamic database components* on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tMysqlConnection</strong> component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Table name</td>
<td>Enter the name of the table.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>COLUMN_NAME: the name of the column currently iterated upon. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DATA_TYPE: the data type of the column currently iterated upon. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>COLUMN_DEFAULT: the default value of the column currently iterated upon. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>IS_NULLABLE: nullability of the column currently iterated upon. This is a Flow variable and it returns YES if the column allows NULL; otherwise, NO.</td>
</tr>
<tr>
<td></td>
<td>COLUMN_KEY: the key indicator of the column currently iterated upon. This is a Flow variable and it returns a string.</td>
</tr>
</tbody>
</table>
**tMysqlColumnList**

- **string if the column is set to Key; otherwise it returns nothing.**

**CHARACTER_MAXIMUM_LENGTH**: the maximum length in number of characters of the column currently iterated upon. This is a Flow variable, and it returns a numeric string for binary or character data types and NULL for other types.

**NUMERIC_PRECISION**: the data precision of the column currently iterated upon. This is a Flow variable, and it returns a numeric string for approximate numeric data, exact numeric data, integer data, or monetary data and NULL for other data types.

**NUMERIC_SCALE**: the scale in digits allowed in the column currently iterated upon. This is a Flow variable, and it returns a numeric string for approximate numeric data, exact numeric data, integer data, or monetary data and NULL for other data types.

**NB_COLUMN**: the number of columns iterated upon so far. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with Mysql components, especially with tMysqlConnection.</th>
</tr>
</thead>
</table>

| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using... |
Scenario: Iterating on a DB table and listing its column names

The following Java scenario creates a five-component job that iterates on a given table name from a Mysql database using a Where clause and lists all column names present in the table.

- Drop the following components from the Palette onto the design workspace: tMysqlConnection, tMysqlTableList, tMysqlColumnList, tFixedFlowInput, and tLogRow.
- Connect tMysqlConnection to tMysqlTableList using an OnSubjobOk link.
- Connect tMysqlTableList, tMysqlColumnList, and tFixedFlowInput using Iterate links.
- Connect tFixedFlowInput to tLogRow using a Row Main link.

- In the design workspace, select tMysqlConnection and click the Component tab to define its basic settings.
- In the Basic settings view, set the database connection details manually or select them from the context variable list, through a Ctrl+Space click in the corresponding field if you have stored them locally as Metadata DB connection entries.

For more information about Metadata, see Talend Studio User Guide.

In this example, we want to connect to a Mysql database called customers.
• In the design workspace, select tMysqlTableList and click the Component tab to define its basic settings.

• On the Component list, select the relevant Mysql connection component if more than one connection is used.

• Enter a Where clause using the right syntax in the corresponding field to iterate on the table name(s) you want to list on the console.

  In this scenario, the table we want to iterate on is called customer.

• In the design workspace, select tMysqlColumnList and click the Component tab to define its basic settings.

• On the Component list, select the relevant Mysql connection component if more than one connection is used.

• In the Table name field, enter the name of the DB table you want to list its column names.

  In this scenario, we want to list the columns present in the DB table called customer.

• In the design workspace, select tFixedFlowInput and click the Component tab to define its basic settings.

• Set the Schema to Built-In and click the three-dot [...] button next to Edit Schema to define the data you want to use as input. In this scenario, the schema is made of two columns, the first for the table name and the second for the column name.
• Click **OK** to close the dialog box, and accept propagating the changes when prompted by the system. The defined columns display in the **Values** panel of the **Basic settings** view.

• Click in the **Value** cell for each of the two defined columns and press **Ctrl+Space** to access the global variable list.

• From the global variable list, select `((String)globalMap.get("tMysqlTableList_1_CURRENT_TABLE"))` and `((String)globalMap.get("tMysqlColumnList_1_COLUMN_NAME"))` for the **TableName** and **ColumnName** respectively.

• In the design workspace, select **tLogRow**.

• Click the **Component** tab and define the basic settings for **tLogRow** as needed.

• Save your job and press **F6** to execute it.

```
Starting job Column_Table_List at 00:55 17/11/2008
customer|id
customer|FirstName
customer|LastName
customer|Address
customer|id_State
Job Column_Table_List ended at 00:55 17/11/2008. [exit code=0]
```

The name of the DB table is displayed on the console along with all its column names.
tMysqlCommit

Commits in one go, using a unique connection, a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

`tMysqlCommit` validates the data processed through the Job into the connected DB.

**tMysqlCommit Standard properties**

These properties are used to configure `tMysqlCommit` running in the Standard Job framework.

The Standard `tMysqlCommit` component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <code>tMysqlConnection</code> component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Warning:**

*If you want to use a Row > Main connection to link `tMysqlCommit` to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.*

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | `ERROR_MESSAGE`: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions |
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

---

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other tMysql* components, especially with the tMysqlConnecti on and tMysqlRollback components.</th>
</tr>
</thead>
</table>

| Dynamic settings | Click the [+ ] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |

---

### Related scenario

For tMysqlCommit related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476.
**tMysqlConnection**

Opens a connection to the specified MySQL database for reuse in the subsequent subjob or subjobs.

**tMysqlConnection Standard properties**

These properties are used to configure tMysqlConnection running in the Standard Job framework.

The Standard tMysqlConnection component belongs to the Databases and the ELT families.

The component in this framework is generally available.

ℹ️ **Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored.</td>
</tr>
<tr>
<td></td>
<td>The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Additional JDBC parameters</strong></td>
<td>Specify additional connection properties for the DB connection you are creating.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Use or register a shared DB Connection</strong></td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the <strong>Shared DB Connection Name</strong> field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.</td>
</tr>
</tbody>
</table>
This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the `tRunJob` component. Using a shared connection together with a `tRunJob` component with either of these two options enabled will cause your Job to fail.

This check box is not available when the **Specify a data source alias** check box is selected.

### Specify a data source alias

Select this check box and specify the alias of a data source created on the **Talend Runtime** side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in **Talend Runtime**. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

### Advanced settings

#### Auto Commit

Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

#### tStatCatcher Statistics

Select this check box to gather the job processing metadata at a Job level as well as at each component level.

### Usage

#### Usage rule

This component is more commonly used with other `tMysql*` components, especially with the `tMysqlCommit` and `tMysqlRollback` components.

### Scenario: Inserting data in mother/daughter tables

The following Job is dedicated to advanced database users, who want to carry out multiple table insertions using a parent table id to feed a child table.

As a prerequisite to this Job, follow the steps described below to create the relevant tables using an engine such as **innodb**:

---

**tMysqlConnection**

This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the `tRunJob` component. Using a shared connection together with a `tRunJob` component with either of these two options enabled will cause your Job to fail.

This check box is not available when the **Specify a data source alias** check box is selected.

**Specify a data source alias**

Select this check box and specify the alias of a data source created on the **Talend Runtime** side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in **Talend Runtime**. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

### Advanced settings

#### Auto Commit

Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

#### tStatCatcher Statistics

Select this check box to gather the job processing metadata at a Job level as well as at each component level.

### Usage

#### Usage rule

This component is more commonly used with other `tMysql*` components, especially with the `tMysqlCommit` and `tMysqlRollback` components.

### Scenario: Inserting data in mother/daughter tables

The following Job is dedicated to advanced database users, who want to carry out multiple table insertions using a parent table id to feed a child table.

As a prerequisite to this Job, follow the steps described below to create the relevant tables using an engine such as **innodb**:

---

**tMysqlConnection**

This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the `tRunJob` component. Using a shared connection together with a `tRunJob` component with either of these two options enabled will cause your Job to fail.

This check box is not available when the **Specify a data source alias** check box is selected.

**Specify a data source alias**

Select this check box and specify the alias of a data source created on the **Talend Runtime** side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in **Talend Runtime**. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

### Advanced settings

#### Auto Commit

Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

#### tStatCatcher Statistics

Select this check box to gather the job processing metadata at a Job level as well as at each component level.

### Usage

#### Usage rule

This component is more commonly used with other `tMysql*` components, especially with the `tMysqlCommit` and `tMysqlRollback` components.

### Scenario: Inserting data in mother/daughter tables

The following Job is dedicated to advanced database users, who want to carry out multiple table insertions using a parent table id to feed a child table.

As a prerequisite to this Job, follow the steps described below to create the relevant tables using an engine such as **innodb**:
1. In a command line editor, connect to your Mysql server.

2. Once connected to the relevant database, type in the following command to create the parent table:
   
   ```sql
   create table f1090_mum(id int not null auto_increment, name varchar(10), primary key(id)) engine=innodb;
   ```

3. Then create the second table:
   
   ```sql
   create table f1090_baby (id_baby int not null, years int) engine=innodb;
   ```

Back in *Talend Studio*, the Job requires seven components including *tMysqlConnection* and *tMysqlCommit*.

**Linking the components**

**Procedure**

1. Drag and drop the following components from the Palette: a *tFileList*, a *tFileInputDelimited*, a *tMap*, a *tMysqlConnection*, a *tMysqlCommit* and two *tMysqlOutput*.

2. Connect *tMysqlConnection* to *tFileList* using an *OnComponentOk* link.

3. Connect *tFileList* to *tMysqlCommit* using an *OnComponentOk* link.

4. Connect the *tFileList* component to the input file component using an *Iterate* link as the name of the file to be processed will be dynamically filled in from the *tFileList* directory using a global variable.

5. Connect the *tFileInputDelimited* component to the *tMap* and dispatch the flow between the two output Mysql DB components. Use a *Row* link for each for these connections representing the main data flow.

**Configuring the components**

**Procedure**

1. Set the *tFileList* component properties, such as the directory name where files will be fetched from.
2. In the **tMysqlConnection** Component view, set the connection details manually or fetch them from the Repository if you centrally stored them as a Metadata DB connection entry. For more information about Metadata, see *Talend Studio User Guide*.

3. On the **tFileInputDelimited** component’s **Basic settings** panel, press Ctrl+Space bar to access the variable list. Set the **File Name** field to the global variable: `tFileList_1.CURRENT_FILEPATH`

4. Set the rest of the fields as usual, defining the row and field separators according to your file structure.

5. Then set the schema manually through the **Edit schema** feature or select the schema from the Repository. In Java version, make sure the data type is correctly set, in accordance with the nature of the data processed.

6. In the **tMap** Output area, add two output tables, one called mum for the parent table, the second called baby, for the child table.
   - Drag the **Name** column from the **Input** area, and drop it to the mum table.
   - Drag the **Years** column from the **Input** area and drop it to the baby table.

7. Make sure the mum table is on the top of the baby table as the order is determining for the flow sequence hence the DB insert to perform correctly.
   - Connect the output row link to distribute correctly the flow to the relevant DB output component.

8. In each of the **tMysqlOutput** components’ **Basic settings** panel, select the **Use an existing connection** check box to retrieve the **tMysqlConnection** details.
9. Set the Table name making sure it corresponds to the correct table, in this example either f1090_mum or f1090_baby.

There is no action on the table as they are already created.

Select Insert as Action on data for both output components.

Click on Sync columns to retrieve the schema set in the tMap.

10. In the Additional columns area of the DB output component corresponding to the child table (f1090_baby), set the id_baby column so that it reuses the id from the parent table.

11. In the SQL expression field type in: "(Select Last_Insert_id())"

The position is Before and the Reference column is years.

In the Advanced settings panel, clear the Extend insert check box.

**Executing the Job**

**Procedure**

1. Press Ctrl + S to save your Job.
2. Press F6 to execute it.

The parent table id has been reused to feed the id_baby column.
Scenario: Rollback from inserting data in mother/daughter tables

About this task

Based on Scenario: Inserting data in mother/daughter tables on page 2476, insert a rollback function in order to prevent unwanted commit.

Procedure

1. Drag and drop tMysqlRollback to the design workspace. tMysqlRollback ensures that the transaction will not be partly committed.
2. Connect tFileList to tMysqlRollback with an OnSubjobError trigger.
3. Double-click tMysqlRollback to enter its Basic settings view.
4. From the Component List, select the connection component.

Scenario: Sharing a database connection between a parent Job and child Job

This scenario shows how a database connection is shared between a parent Job and a child Job. The parent Job first calls the child Job to write some randomly generated data to a MySQL database, and then reads the data from the MySQL database and displays it on the console. A connection to the MySQL database is set up only once and used in both Jobs.

Setting up the child Job

Adding and linking components

Procedure

1. Add the following components by typing the component names on the design workspace or dropping them from the Palette:
   - a tMysqlConnection component, to open a connection to the MySQL database,
   - a tRowGenerator, to generate random input data,
• a tMysqlOutput component, to write data to the MySQL database.

2. Connect the tRowGenerator component to the tMysqlOutput component using a Row > Main connection.

3. Connect the tMysqlConnection component to the tRowGenerator using a Trigger > OnSubjobOk connection.

Configuring the database connection

Procedure

1. Double-click the tMysqlConnection component to open its Basic settings view.

2. With the Property Type set to Built-In, specify the connection details in the relevant fields, including:
   • the host name or IP address of your database server
   • the listening port number
   • the database name
   • the user name and password for your database authentication.

   If you have stored your connection details under the Metadata node in the Repository tree view, you can simply drop your centralized metadata item onto the tMysqlConnection component. For
information on how to centralize a database connection, see the chapter on managing metadata of the *Talend Studio User Guide*.

3. Select the **Use or register a shared DB Connection** check box so that the database connection open by this component can be shared across different Jobs. Then, enter a name for the shared connection between double quotes in the **Shared DB Connection Name** field, *shared_mysql_connection* in this example.

### Configuring the input data

**Procedure**

1. Double-click the **tRowGenerator** component to open the Row Generator editor.

2. Click the [+ ] button to add four columns and set their properties:
   - **id**, type Integer, 2 characters long.
   - **firstName**, type String, 15 characters long
   - **lastName**, type String, 15 characters long
   - **city**, type String, 15 characters long

3. Define the function for each column:
   - For the **id** column, select **Numeric.sequence** from the **Function** list to generate sequence numbers.
   - For the **firstName** column, select **TalendDataGenerator.getFirstName** from the **Function** list to generate random first names.
   - For the **lastName** column, select **TalendDataGenerator.getLastName** from the **Function** list to generate random family names.
• For the city column, select `TalendDataGenerator.getUsCity` from the Function list to generate random city names.

4. In the Number of Rows for RowGenerator field, specify the number of data rows you want to generate, 10 in this example.

5. Click the Preview button on the Preview tab to validate the generator settings.

6. When done, click OK to close the editor and click Yes when prompted to propagate the schema to the next component.

Configuring the database output

Procedure

1. Double-click the tMysqlOutput component to open its Basic settings view.

   ![tMysqlOutput component settings](image)

   - Select the Use an existing connection check box and, in case you have more than one connection component in the Job, select the connection component to be used from the Component List dropdown list.
   - In the Table field, enter the name of the database table you are going to write data to, customers in this example.
   - From the Action on table list, select the Drop table if exists and create option to ensure a clean table is created.
   - From the Action on data list, select Insert.
   - Click the [...] button next to Edit schema to check the output schema. If needed, click Sync columns to retrieve the schema from the preceding component.
   - Press Ctrl+S to save the Job.

Setting up the parent Job

Arranging data flow for the parent Job

Procedure

1. Add the following components by typing the component names on the design workspace or dropping them from the Palette:
   - a tRunJob component, to call the child Job,
   - a tMysqlConnection component, to open the MySQL database connection,
   - a tMysqlInput component, to read the data written to the MySQL database by the child Job.
   - a tLogRow component to display the data on the console,
• a **tMysqlCommit** component to commit data upon transation and close the database connection.

2. Connect the **tRunJob** component to the **tMysqlConnection** component using a **Trigger > OnSubjobOk** connection.

3. Connect the **tMysqlConnection** component to the **tMysqlInput** component using a **Trigger > OnSubjobOk** connection.

4. Connect the **tMysqlInput** component to the **tLogRow** component using a **Row > Main** connection.

5. Connect the **tMysqlInput** component to the **tMysqlCommit** component using a **Trigger > OnSubjobOk** connection.

---

### Configuring the components

#### Procedure

1. Double-click the **tRunJob** component to open its **Basic settings** view.
   
   If a child Job has been already specified in the component, you can open its **Basic settings** view by right-clicking it and selecting **Settings** from the contextual menu, or by clicking the component and then selecting the **Component** tab.

   ![tRunJob component settings](image)

2. Click the [...] button next to the **Job** field and select the child Job from the [Repository Content] dialog box. Leave all the other parameters as default.
3. In the Basic settings view of the tMysqlConnection component, select the **Use or register a shared DB Connection** check box. Then, in the **Shared DB Connection Name** field, enter the name of the shared connection defined in the child Job, `shared_mysql_connection` in this example. Leave all the other parameters blank or as default.

![tMysqlConnection Component](image)

4. Double-click the tMysqlInput component to open its Basic settings view.

![tMysqlInput Component](image)

5. Select the **Use an existing connection** check box and, in case you have more than one connection component in the Job, select the connection component to be used from the **Component List** drop-down list.

6. Click the [...] button next to **Edit schema** to open the [Schema] editor, and define the same data structure as in the child Job.

![Schema Editor](image)
When done, click **OK** to close the dialog and click **Yes** when prompted to propagate the schema to the next component.

7. Specify the table name in the **Table Name** list, and click **Guess Query** to get the query statement automatically filled in the **Query** field.

8. In the **Basic settings** view of the **tLogRow** component, select the **Table** mode to display the execution result in table cells.

   Leave the settings of the **tMysqlCommit** component unchanged.

**Execute the Job**

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** or click the **Run** button on the **Run** console to execute the Job.
The Job first calls the child Job to write data to the database and then reads the data from the database and displays it on the console.
tMysqlInput

Executes a DB query with a strictly defined order which must correspond to the schema definition. tMysqlInput reads a database and extracts fields based on a query. Then it passes on the field list to the next component via a Main row link.

tMysqlInput Standard properties

These properties are used to configure tMysqlInput running in the Standard Job framework. The Standard tMysqlInput component belongs to the Databases family. The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-In or Repository.</td>
</tr>
<tr>
<td>Built-In: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file where the properties are stored.</td>
<td></td>
</tr>
</tbody>
</table>

Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

| DB version | Select the version of the database to be used. |
| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. |

Note:
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component.
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see *Talend Studio User Guide.*

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide.</em></td>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide.</em></td>
</tr>
<tr>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
<td></td>
</tr>
<tr>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the <strong>[Repository Content]</strong> window.</td>
<td></td>
</tr>
<tr>
<td><strong>Table Name</strong></td>
<td>Name of the table to be read.</td>
</tr>
<tr>
<td><strong>Query type and Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
</tbody>
</table>
### Specify a data source alias

Select this check box and specify the alias of a data source created on the *Talend Runtime* side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in *Talend Runtime*. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

**Warning:**

If you use the component’s own DB configuration, your data source connection will be closed at the end of the component. To prevent this from happening, use a shared DB connection with the data source alias specified.

This check box is not available when the *Use an existing connection* check box is selected.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Advanced settings</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Specify a data source alias</strong></td>
<td>Select this check box and specify the alias of a data source created on the <em>Talend Runtime</em> side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in <em>Talend Runtime</em>. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503. <strong>Warning:</strong> If you use the component’s own DB configuration, your data source connection will be closed at the end of the component. To prevent this from happening, use a shared DB connection with the data source alias specified. This check box is not available when the <em>Use an existing connection</em> check box is selected.</td>
</tr>
<tr>
<td><strong>Additional JDBC parameters</strong></td>
<td>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <em>Use an existing connection</em> check box in the <strong>Basic settings</strong>. <strong>Note:</strong> When you need to handle data of the time-stamp type <em>0000-00-00 00:00:00</em> using this component, set the parameter as: <code>noDatetimeStringSync=true&amp;zeroDateTimeBehavior=convertToNull</code>.</td>
</tr>
<tr>
<td><strong>Enable stream</strong></td>
<td>Select this check box to enables streaming over buffering which allows the code to read from a large table without consuming a large amount of memory in order to optimize the performance. This check box is available only when Mysql 4 or Mysql 5 is selected from the <strong>DB Version</strong> drop-down list.</td>
</tr>
<tr>
<td><strong>Trim all the String/Char columns</strong></td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns. <strong>Note:</strong> Clear <strong>Trim all the String/Char columns</strong> to enable Trim columns in this field.</td>
</tr>
<tr>
<td><strong>Trim column</strong></td>
<td>Remove leading and trailing whitespace from defined columns. <strong>Note:</strong> Clear <strong>Trim all the String/Char columns</strong> to enable Trim columns in this field.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>QUERY: the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible SQL queries for Mysql databases.</th>
</tr>
</thead>
</table>

#### Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.
Scenario: Writing columns from a MySQL database to an output file using tMysqlInput

In this scenario we will read certain columns from a MySQL database, and then write them to a table in a local output file.

Dragging and dropping components and linking them together

Procedure
1. Drop tMysqlInput and tFileOutputDelimited from the Palette onto the workspace.
2. Link tMysqlInput to tFileOutputDelimited using a Row > Main connection.

Configuring the components

Procedure
1. Double-click tMysqlInput to open its Basic Settings view in the Component tab.

2. From the Property Type list, select Repository if you have already stored the connection to database in the Metadata node of the Repository tree view. The property fields that follow are automatically filled in.

For more information about how to store a database connection, see Talend Studio User Guide.

If you have not defined the database connection locally in the Repository, fill in the details manually after selecting Built-in from the Property Type list.

3. Set the Schema as Built-in and click Edit schema to define the desired schema.

The schema editor opens:
4. Click the [+] button to add the rows that you will use to define the schema, four columns in this example: id, first_name, city and salary.
   Under Column, click in the fields to enter the corresponding column names.
   Click the field under Type to define the type of data.
   Click OK to close the schema editor.

5. Next to the Table Name field, click the [...] button to select the database table of interest.
   A dialog box displays a tree diagram of all the tables in the selected database:

6. Click the table of interest and then click OK to close the dialog box.

7. Set the Query Type as Built-In.

8. In the Query box, enter the query required to retrieve the desired columns from the table.

9. Double-click tFileOutputDelimited to set its Basic settings in the Component tab.
Next to the **File Name** field, click the [...] button to browse your directory to where you want to save the output file, then enter a name for the file.

Select the **Include Header** check box to retrieve the column names as well as the data.

**Save the Job.**

**Executing the Job**

The results below can be found after **F6** is pressed to run the Job.

<table>
<thead>
<tr>
<th></th>
<th>id;first_name;city;salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1;Martin;Sacramento;9011</td>
</tr>
<tr>
<td>2</td>
<td>2;Zachary;Atlanta;8118</td>
</tr>
<tr>
<td>3</td>
<td>3;James;Hartford;5087</td>
</tr>
<tr>
<td>4</td>
<td>4;Herbert;Charleston;9233</td>
</tr>
<tr>
<td>5</td>
<td>5;Herbert;Hartford;5289</td>
</tr>
<tr>
<td>6</td>
<td>6;Ulysses;Nashville;6269</td>
</tr>
</tbody>
</table>

As shown above, the output file is written with the desired column names and corresponding data, retrieved from the database:

**Note:**

The Job can also be run in the **Traces Debug** mode, which allows you to view the rows as they are being written to the output file, in the workspace.

**Scenario: Using context parameters when reading a table from a database**

In this scenario, MySQL is used for demonstration purposes. We will read a table from a MySQL database, using a context parameter to refer to the table name.

**Dragging and dropping components and linking them together**

**Procedure**

1. Drop **tMysqlInput** and **tLogRow** from the **Palette** onto the workspace.
2. Link **tMysqlInput** to **tLogRow** using a **Row > Main** connection.
Configuring the components

Procedure

1. Double-click tMysqlInput to open its Basic Settings view in the Component tab.

2. From the Property Type list, select Repository if you have already stored the connection to database in the Metadata node of the Repository tree view. The property fields that follow are automatically filled in.

   For more information about how to store a database connection, see Talend Studio User Guide.

   If you have not defined the database connection in the Repository, fill in the details manually after selecting Built-in from the Property Type list.

3. Set the Schema as Built-In and click Edit schema to define the desired schema.

   The schema editor opens:
4. Click the [+ ] button to add the rows that you will use to define the schema, seven columns in this example:  \textit{id, first\_name, last\_name, city, state, date\_of\_birth and salary}. 
Under Column, click the fields to enter the corresponding column names. 
Click the fields under Type to define the type of data. 
Click OK to close the schema editor.

5. Put the cursor in the Table Name field and press F5 for context parameter setting.

![Context Parameter](image)

For more information about context settings, see \textit{Talend Studio User Guide}.

6. Keep the default setting in the Name field and type in the name of the database table in the Default value field, employees in this case.

7. Click Finish to validate the setting. 
The context parameter context\_TABLE automatically appears in the Table Name field.

8. In the Query type list, select Built-In. Then, click Guess Query to get the query statement.
In this use case, we want to read the records with the salary above 8000. Therefore, we add a Where clause and the final query statement is as follows:

```
"SELECT  
"+context.TABLE+".`id`,  
"+context.TABLE+".`first\_name`,  
"+context.TABLE+".`last\_name`,  
"+context.TABLE+".`city`,  
"+context.TABLE+".`state`,  
"+context.TABLE+".`date\_of\_birth`,  
"+context.TABLE+".`salary`  
FROM "+context.TABLE+"  
WHERE  
"+context.TABLE+".`salary` > 8000"
```

9. Double-click tLogRow to set its Basic Settings in the Component tab.
10. In the **Mode** area, select **Table (print values in cells of a table)** for a better display of the results.

11. Save the Job.

**Executing the Job**

The results below can be found after **F6** is pressed to run the Job.

<table>
<thead>
<tr>
<th>id</th>
<th>first_name</th>
<th>last_name</th>
<th>city</th>
<th>state</th>
<th>date_of_birth</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Martin</td>
<td>Roosevelt</td>
<td>Sacramento</td>
<td>ND</td>
<td>19-10-1950</td>
<td>9011</td>
</tr>
<tr>
<td>2</td>
<td>Zachary</td>
<td>Johnson</td>
<td>Atlanta</td>
<td>ND</td>
<td>14-11-1978</td>
<td>8118</td>
</tr>
<tr>
<td>4</td>
<td>Herbert</td>
<td>Harrison</td>
<td>Charleston</td>
<td>AE</td>
<td>15-08-1963</td>
<td>9233</td>
</tr>
</tbody>
</table>

As shown above, the records with the salary greater than 8000 are retrieved.

**Scenario: Reading data from databases through context-based dynamic connections**

In this scenario, MySQL is used for demonstration purposes. We will read data from database tables with the same data structure but in two different MySQL databases named **project_q1** and **project_q2** respectively. We will specify the connections to these databases dynamically at runtime, without making any modification to the Job.

**Dropping and linking the components**

**Procedure**

1. Drop two **tMysqlConnection**, a **tMysqlInput**, a **tLogRow**, and a **tMysqlClose** components onto the design workspace.

2. Link the first **tMysqlConnection** to the second **tMysqlConnection** and the second **tMysqlConnection** to **tMysqlInput** using **Trigger > On Subjob Ok** connections.

3. Link **tMysqlInput** to **tLogRow** using a **Row > Main** connection.

4. Link **tMysqlInput** to **tMysqlClose** using a **Trigger > On Subjob Ok** connection.
Creating a context variable

About this task

To be able to choose a database connection dynamically at runtime, we need to define a context variable, which will then be configure it in the Dynamic settings of the database input component.

Procedure

1. In the Contexts view, click the `[+]` button to add a row in the table, click in the Name field and enter a name for the variable, myConnection in this example.

2. From the Type list field, select List Of Value.
3. Click in the Value field and then click the button that appears in the field to open the [Configure value of list] dialog box.
4. In the [Configure value of list] dialog box, click the New... button to open the [New Value] dialog box, and enter the name of one of the connection components in the text field, tMysqlConnection_1 in this example. Then click OK to close the dialog box.

Repeat this step to specify the other connection component name as another list item, tMysqlConnection_2 in this example.

When done, click OK to close the [Configure Values] dialog box.

5. Select the check box next to the variable value field, and fill the Prompt field with the message you want to display at runtime, Select a connection component: in this example.
Configuring the components

Procedure

1. Double-click the first tMysqlConnection component to show its Basic settings view, and set the connection details. For more information on the configuration of tMysqlConnection, see tMysqlConnection on page 2475.

   Note that we use this component to open a connection to a MySQL database named project_q1.

2. Configure the second tMysqlConnection component in the same way, but fill the Database field with project_q2 because we want to use this component to open a connection to another MySQL database, project_q2.

3. Double-click the tMysqlInput component to show its Basic settings view.
4. Select the **Use an existing connection** check box, and leave the **Component List** box as it is.

5. Click the [...] button next to **Edit schema** to open the **[Schema]** dialog box and define the data structure of the database table to read data from.

![Schema of tMysqlInput_1](image)

In this example, the database table structure is made of four columns, *id* (type Integer, 2 characters long), *firstName* (type String, 15 characters long), *lastName* (type String, 15 characters long), and *city* (type String, 15 characters long). When done, click **OK** to close the dialog box and propagate the schema settings to the next component.

6. Fill the **Table** field with the database table name, *customers* in this example, and click **Guess Query** to generate the query statement corresponding to your table schema in the **Query** field.

7. In the **Dynamic settings** view, click the [+ ] button to add a row in the table, and fill the **Code** field with the code script of the context variable you just created, " + context.myConnection + " in this example.

![tMysqlInput_1](image)

8. In the **Basic settings** view of the **tLogRow** component, select the **Table** option for better display effect of the Job execution result.
9. In the **Dynamic settings** view of the **tMysqlClose** component, do exactly the same as in the **Dynamic settings** view of the **tMysqlInput** component.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job and press **F6** or click **Run** to launch it.

   A dialog box appears prompting you to specify the connection component you want to use.

2. Select the connection component, **tMysqlConnection_1**, and click **OK**.

   The data read from database *project_q1* is displayed in the **Run** console.
3. Press F6 or click Run to launch your Job again. When prompted, select the other connection component, **tMysqlConnection_2**, to read data from the other database, **project_q2**.

The data read from database **project_q2** is displayed in the Run console.

---

**Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database**

This scenario describes a two-component Job that retrieves data from a MySQL database table and displays the data on the console. The Job will be deployed in Talend Runtime and will use the data source created on Talend Runtime to connect to the database defined in a Runtime configuration file.

**Creating a MySQL data source in Talend Runtime Container**

To be able to retrieve data from a MySQL database table and displays the data on the console, you must have created a MySQL data source in Talend Runtime Container.

**Before you begin**

You must have properly installed Talend Runtime in the machine in which you want to execute your Job. For further information, search for installing Talend Runtime on Talend Help Center (https://help.talend.com).

**Procedure**

1. Install the MySQL JDBC driver by running the following `bundle:install` command in Talend Runtime Container:
2. Install the database connection pool by running the following `bundle:install` command in Talend Runtime Container:

```
karaf@trun()> bundle:install mvn:mysql/mysql-connector-java/5.1.18
```

3. Copy the data source configuration file `datasource-mysql.xml` under the folder `<TalendRuntimePath>/add-ons/datasources/dataservice` to the folder `<TalendRuntimePath>/container/deploy` and then in the file update the username and password property values required to connect to your database.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
  default-activation="lazy">
  <bean id="mysqlDataSource" class="com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource">
    <property name="url" value="jdbc:mysql://localhost:3306/test"/>
    <property name="user" value="root"/>
    <property name="password" value="talend"/>
  </bean>

  <bean id="dataSource" class="org.apache.commons.dbcp.datasources.SharedPoolDataSource" destroy-method="close">
    <property name="connectionPoolDataSource" ref="mysqlDataSource"/>
    <property name="maxActive" value="20"/>
    <property name="maxIdle" value="5"/>
    <property name="maxWait" value="-1"/>
  </bean>

  <service ref="dataSource" interface="javax.sql.DataSource">
    <service-properties>
      <entry key="osgi.jndi.service.name" value="jdbc/sample"/>
    </service-properties>
  </service>
</blueprint>
```

**Note:** The property `osgi.jndi.service.name` in the data source configuration file defines the data source alias, `jdbc/sample`, in this example. The alias will be used in the database component to connect to the database defined in the data source configuration file when the Job is deployed in Talend Runtime.

For more information about how to use the data source in Talend Runtime Container, see *Talend ESB Container Administration Guide* and *Talend ESB Studio User Guide*.

### Adding and linking the components

**Procedure**

1. Create a new Job and add a `tMysqlInput` component and a `tLogRow` component by typing their names in the design workspace or dropping them from the Palette.

2. Link the `tMysqlInput` component to the `tLogRow` component using a `Row > Main` connection.
Configuring the components

Procedure

1. Double-click tMysqlInput to open its Basic Settings view in the Component tab.

![tMysqlInput settings](image)

- **Property Type**: Built-In
- **DB Version**: Mysql 5
- **Use an existing connection**: unchecked
- **Host**: unspecified
- **Port**: 3306
- **Database**: unspecified
- **Username**: unspecified
- **Password**: unspecified
- **Scheme**: Built-In
- **Table Name**: student
- **Query Type**: `SELECT
student.sid, 
student.sname
FROM student`
- **Data source alias**: jdbc/sample

2. In the Table Name field, enter the name of the table from which the data will be retrieved. In this example, it is student.

3. Click the [...] button next to Edit schema to open the Schema dialog box and define the schema of the table student by adding two columns: sid of Integer type and sname of String type.

![Schema dialog box](image)

4. Click the Guess Query button to fill the Query field with the SQL query statement to be used to retrieve data from the specified table.

5. Select the Specify a data source alias check box and in the Data source alias field displayed enter the data source alias specified in the data source configuration file datasource-mysql.xml on Talend Runtime side. In this example, it is jdbc/sample.

6. Double-click the tLogRow component to open its Basic settings view.
7. In the Mode area, select **Table (print values in cells of a table)** for better readability of the result.

**Building the Job and deploying it in Talend Runtime Container**

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Start the Talend Runtime Container.
3. In the Repository tree view of your Talend Studio, right-click the Job and from the contextual menu select **Build Job** to open the [Build Job] dialog box.

4. Select **OSGI Bundle For ESB** from the **Select the build type** drop-down list.
5. Click the **Browse** button next to the *To archive file* field to specify where you will store the Job archive file. In this example, we will export the Job directly to the deploy folder `<TalendRuntime Path>/container/deploy` so that the Job can be run immediately in Talend Runtime after being exported.

For more information about how to build a Job, see *Talend Studio User Guide*.

6. Click **Finish** to close the [Build Job] dialog box.

As you can see on the Talend Runtime Container console, the Job starts to run immediately after being deployed and the data in the *student* table is retrieved and displayed on the console.
tMysqlLastInsertId

Obtains the primary key value of the record that was last inserted in a Mysql table by a user.
tMysqlLastInsertId fetches the last inserted ID from a selected MySQL Connection.

**tMysqlLastInsertId Standard properties**

These properties are used to configure tMysqlLastInsertId running in the Standard Job framework.
The Standard tMysqlLastInsertId component belongs to the Databases family.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related
to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components
on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema</strong> and <strong>Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
</tbody>
</table>

**Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.
Component list

Select the relevant **tMysqlConnection** component in the list if more than one connection is planned for the current job.

### Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

### Global Variables

#### Global Variables

**NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

#### Usage rule

This component is to be used as an intermediary component.

**Warning:**

*If you use this component with **tMySqlOutput**, verify that the **Extend Insert** check box in the **Advanced Settings** tab is not selected. **Extend Insert** allows you to make a batch insertion, however, if the check box is selected, only the ID of the last line in the last batch will be returned.*

#### Dynamic settings

Click the [*] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.
Scenario: Get the ID for the last inserted record with tMysqlLastInsertId

The following Java scenario creates a job that opens a connection to Mysql database, writes the defined data into the database, and finally fetches the last inserted ID on the existing connection.

- Drop the following components from the Palette onto the design workspace: tMySqlConnection, tMySqlCommit, tFileInputDelimited, tMySqlOutput, tMysqlLastInsertId, and tLogRow.

- Connect tMySqlConnection to tFileInputDelimited using an OnSubjobOk link.

- Connect tFileInputDelimited to tMySqlCommit using an OnSubjobOk link.

- Connect tFileInputDelimited to the three other components using Row Main links.

- In the design workspace, select tMysqlConnection.

- Click the Component tab to define the basic settings for tMysqlConnection.

- In the Basic settings view, set the connection details manually or select them from the context variable list, through a Ctrl+Space click in the corresponding field if you stored them locally as Metadata DB connection entries. For more information about Metadata, see Talend Studio User Guide.
In the design workspace, select **tMysqlCommit** and click the **Component** tab to define its basic settings.

On the **Component List**, select the relevant **tMysqlConnection** if more than one connection is used.

In the design workspace, select **tFileInputDelimited**.

Click the **Component** tab to define the basic settings of **tFileInputDelimited**.

- Fill in a path to the processed file in the **File Name** field. The file used in this example is **Customers**.
- Define the **Row separator** that allow to identify the end of a row. Then define the **Field separator** used to delimit fields in a row.
- Set the header, the footer and the number of processed rows as necessary. In this scenario, we have one header.
- Click the three-dot button next to **Edit Schema** to define the data to pass on to the next component.

In this scenario, the schema consists of two columns, *name* and *age*. The first holds three employees’ names and the second holds the corresponding age for each.

- In the design workspace, select **tMySqlOutput**.
- Click the **Component** tab to define the basic settings of **tMySqlOutput**.
  - Select the **Use an existing connection** check box.
  - In the **Table** field, enter the name of the table where to write the employees’ list, in this example: *employee*.
  - Select relevant actions on the **Action on table** and **Action on data** lists. In this example, no action is carried out on table, and the action carried out on data is **Insert**.
  - Click **Sync columns** to synchronize columns with the previous component. In this example, the schema to be inserted into the MySql database table consists of the two columns *name* and *age*.

- In the design workspace, select **tMySqlLastInsertId**.
• Click the **Component** tab to define the basic settings of **tMysqlLastInsertId**.

![Image of tMysqlLastInsertId_1 component]

- **Basic settings**
- **Advanced settings**
- **Dynamics settings**
- **View**
- **Documentation**

- **Schema of tMysqlLastInsertId_1**

<table>
<thead>
<tr>
<th>Column</th>
<th>Db Column</th>
<th>Key</th>
<th>Type</th>
<th>Nulla</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>name</td>
<td></td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>age</td>
<td>age</td>
<td></td>
<td>Integer</td>
<td></td>
</tr>
</tbody>
</table>

You can select the data type **Long** from the **Type** drop-down list in case of a huge number of entries.

- In the design workspace, select **tLogRow** and click the **Component** tab to define its basic settings. For more information, see **tLogRow** on page 1988.

- Save your job and press **F6** to execute it.

```plaintext
Starting job lastinserted at 11:12 01/09/2008.
<table>
<thead>
<tr>
<th>tLogRow_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
</tr>
<tr>
<td>Marie</td>
</tr>
<tr>
<td>Patrick</td>
</tr>
<tr>
<td>Pierrick</td>
</tr>
</tbody>
</table>
```

**Job lastinserted ended at 11:12 01/09/2008. [exit code=0]**

**tMysqlLastInsertId** fetched the last inserted ID for each line on the existing connection.
tMysqlLookupInput

Reads a MySQL database and extracts fields based on a query.

tMysqllookupInput executes a database query with a strictly defined order which must correspond to the schema definition.

This component also allows you to connect and read data from a RDS Aurora or a RDS MySQL database.
tMysqlOutput

Writes, updates, makes changes or suppresses entries in a database.

tMysqlOutput executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

**tMysqlOutput Standard properties**

These properties are used to configure tMysqlOutput running in the Standard Job framework. The Standard tMysqlOutput component belongs to the Databases family.

The component in this framework is generally available.

ℹ️ **Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property type</strong></td>
<td>Either Built-In or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where the</td>
</tr>
<tr>
<td></td>
<td>properties are stored.</td>
</tr>
<tr>
<td><strong>DB Version</strong></td>
<td>Select the MySQL version you are using.</td>
</tr>
</tbody>
</table>

🔍 Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

ℹ️ **Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component.
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](#).

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
</tbody>
</table>
| Action on table | On the table defined, you can perform one of the following operations:  
| Default: No operation is carried out.  
| Drop and create a table: The table is removed and created again.  
| Create a table: The table does not exist and gets created.  
| Create a table if not exists: The table is created if it does not exist.  
| Drop a table if exists and create: The table is removed if it already exists and created again.  
| Clear a table: The table content is deleted.  
| Truncate table: The table content is quickly deleted. However, you will not be able to rollback the operation. |
| Action on data | On the data of the table defined, you can perform:  
| Insert: Add new entries to the table. If duplicates are found, the job stops.  
| Update: Make changes to existing entries.  
| Insert or update: Insert a new record. If the record with the given reference already exists, an update would be made.  
| Update or insert: Update the record with the given reference. If the record does not exist, a new record would be inserted.  
| Delete: Remove entries corresponding to the input flow.  
| Replace: Add new entries to the table. If an old row in the table has the same value as a new row for a |
**Insert or update on duplicate key or unique index:** Add entries if the inserted value does not exist or update entries if the inserted value already exists and there is a risk of violating a unique index or primary key.

**Insert Ignore:** Add only new rows to prevent duplicate key errors.

**Warning:**

You must specify at least one column as a primary key on which the **Update** and **Delete** operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the update and delete operations. To do that: Select the **Use field options** check box and then in the **Key in update** column, select the check boxes next to the column name on which you want to base the update operation. Do the same in the **Key in delete** column for the deletion operation.

---

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields.

### Built-In

You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.

### Repository

You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in **Talend Help Center** ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the
Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Die on error</th>
</tr>
</thead>
<tbody>
<tr>
<td>This check box is selected by default. Clear the check box to skip the row in error and complete the process for error-free rows. If needed, you can retrieve the rows in error via a Row &gt; Rejects link.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specify a data source alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.</td>
</tr>
</tbody>
</table>

**Warning:**

If you use the component’s own DB configuration, your data source connection will be closed at the end of the component. To prevent this from happening, use a shared DB connection with the data source alias specified.

This check box is not available when the **Use an existing connection** check box is selected.

---

### Advanced settings

<table>
<thead>
<tr>
<th>Additional JDBC parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <strong>Use an existing connection</strong> check box in the Basic settings.</td>
</tr>
</tbody>
</table>

**Note:**

You can press **Ctrl+Space** to access a list of predefined global variables.

<table>
<thead>
<tr>
<th>Extend Insert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to carry out a bulk insert of a defined set of lines instead of inserting lines one by one. The gain in system performance is considerable.</td>
</tr>
</tbody>
</table>

**Number of rows per insert:** enter the number of rows to be inserted per operation. Note that the higher the value specified, the lower performance levels shall be due to the increase in memory demands.

**Note:**

This option is not compatible with the Reject link. You should therefore clear the check box if you are using a Row > Rejects link with this component.

**Warning:**

*If you are using this component with tMysqlLastInsertID, ensure that the Extend Insert check box in Advanced Settings is not selected.*
**Extend Insert** allows for batch loading, however, if the check box is selected, only the ID of the last line of the last batch will be returned.

| **Use Batch** | Select this check box to activate the batch mode for data processing.  
**Note:**  
This check box is available only when you have selected, the Update or the Delete option in the Action on data field. |
| **Batch Size** | Specify the number of records to be processed in each batch.  
This field appears only when the **Use batch mode** check box is selected. |
| **Commit every** | Number of rows to be included in the batch before it is committed to the DB. This option ensures transaction quality (but not rollback) and, above all, a higher performance level. |
| **Additional Columns** | This option is not available if you have just created the DB table (even if you delete it beforehand). This option allows you to call SQL functions to perform actions on columns, provided that these are not insert, update or delete actions, or actions that require pre-processing.  
**Name:** Type in the name of the schema column to be altered or inserted.  
**SQL expression:** Type in the SQL statement to be executed in order to alter or insert the data in the corresponding column.  
**Position:** Select Before, Replace or After, depending on the action to be performed on the reference column.  
**Reference column:** Type in a reference column that tMySqlOutput can use to locate or replace the new column, or the column to be modified. |
| **Use field options** | Select this check box to customize a request, particularly if multiple actions are being carried out on the data. |
| **Use Hint Options** | Select this check box to activate the hint configuration area which helps you optimize a query’s execution. In this area, parameters are:  
- **HINT:** specify the hint you need, using the syntax /*+ */.  
- **POSITION:** specify where you put the hint in a SQL statement.  
- **SQL STMT:** select the SQL statement you need to use. |
| **Enable debug mode** | Select this check box to display each step involved in the process of writing data in the database. |
Use duplicate key update mode insert

| Use duplicate key update mode insert | Updates the values of the columns specified, in the event of duplicate primary keys:
| | **Column**: Between double quotation marks, enter the name of the column to be updated.
| | **Value**: Enter the action you want to carry out on the column.
| | ![Note]
| | To use this option you must first of all select the Insert mode in the Action on data list found in the Basic Settings view.

**tStatCatcher Statistics**

| tStatCatcher Statistics | Select this check box to collect log data at the component level.

**Global Variables**

| Global Variables | 
| | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.
| | **NB_LINE_UPDATED**: the number of rows updated. This is an After variable and it returns an integer.
| | **NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.
| | **NB_LINE_DELETED**: the number of rows deleted. This is an After variable and it returns an integer.
| | **NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer.
| | **QUERY**: the query statement processed. This is an After variable and it returns a string.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

**Usage**

| Usage rule | This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.
| | This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a MySQL database. It also allows you to create a reject flow using a Row > Rejects link to filter data in error. For an example of tMySqlOutput in use, see Scenario: Retrieving data in error with a Reject link on page 2529.

---

2520
Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Scenario: Inserting a column and altering data using tMysqlOutput

This Java scenario is a three-component job that aims at creating random data using a tRowGenerator, duplicating a column to be altered using the tMap component, and eventually altering the data to be inserted based on an SQL expression using the tMysqlOutput component.

- Drop the following components from the Palette onto the design workspace: tRowGenerator, tMap and tMySQLOutput.
- Connect tRowGenerator, tMap, and tMysqlOutput using the Row Main link.
- In the design workspace, select tRowGenerator to display its Basic settings view.
• Click the **Edit schema** three-dot button to define the data to pass on to the **tMap** component, two columns in this scenario, *name* and *random_date*.

![tRowGenerator_1](image1)

• Click **OK** to close the dialog box.

• Click the **RowGenerator Editor** three-dot button to open the editor and define the data to be generated.

![Schema of tRowGenerator_1](image2)

• Click in the corresponding **Functions** fields and select a function for each of the two columns, *getFirstName* for the first column and *getrandomDate* for the second column.

• In the **Number of Rows for Rowgenerator** field, enter 10 to generate ten first name rows and click **Ok** to close the editor.

• Double-click the **tMap** component to open the Map editor. The Map editor opens displaying the input metadata of the **tRowGenerator** component.
In the Schema editor panel of the Map editor, click the plus button of the output table to add two rows and define the first as `random_date` and the second as `random_date1`.

In this scenario, we want to duplicate the `random_date` column and adapt the schema in order to alter the data in the output component.

- In the Map editor, drag the `random_date` row from the input table to the `random_date` and `random_date1` rows in the output table.
• Click **OK** to close the editor.

• In the design workspace, double-click the **tMysqlOutput** component to display its **Basic settings** view and set its parameters.

  ![tMysqlOutput component](image)

  **Dates(tMysqlOutput_1)**

  **Basic settings**
  - **Property Type**: Repository
  - **DB Version**: Mysql
  - **Use an existing connection**
  - **Host**: localhost
  - **Port**: 3306
  - **Database**: test
  - **Username**: root
  - **Password**: ********
  - **Table**: Dates
  - **Action on table**: Drop table if exists and create
  - **Action on data**: Insert
  - **Schema**: Built-In
  - **Data source**: Built-In
  - **This option only applies when deploying and running in the Talend Runtime**
  - **Specify a data source alias**
  - **Die on error**

  ![Schema editor](image)

• Set **Property Type** to Repository and then click the three-dot button to open the [Repository content] dialog box and select the correct DB connection. The connection details display automatically in the corresponding fields.

  **Note:**
  
  If you have not stored the DB connection details in the Metadata entry in the Repository, select **Built-in** on the property type list and set the connection detail manually.

• Click the three-dot button next to the **Table** field and select the table to be altered, Dates in this scenario.
• On the **Action on table** list, select **Drop table if exists and create**, select **Insert** on the **Action on data** list.

• If needed, click **Sync columns** to synchronize with the columns coming from the **tMap** component.

• Click the **Advanced settings** tab to display the corresponding view and set the advanced parameters.

![Image of Dates(tMysqlOutput_1) window](image)

- In the **Additional Columns** area, set the alteration to be performed on columns.

  In this scenario, the *One_month_later* column replaces *random_date_1*. Also, the data itself gets altered using an SQL expression that adds one month to the randomly picked-up date of the *random_date_1* column. ex: 2007-08-12 becomes 2007-09-12.

  - Enter *One_Month_Later* in the **Name** cell.
  - In the **SQL expression** cell, enter the relevant addition script to be performed, "adddate(Rando m_date, interval 1 month)" in this scenario.
  - Select **Replace** on the **Position** list.
  - Enter *Random_date1* on the **Reference column** list.

**Note:**

For this job we duplicated the *random_date_1* column in the DB table before replacing one instance of it with the *One_Month_Later* column. The aim of this workaround was to be able to view upfront the modification performed.

• Save your job and press **F6** to execute it.

The new *One_month_later* column replaces the *random_date1* column in the DB table and adds one month to each of the randomly generated dates.
Scenario: Updating data using tMysqlOutput

This Java scenario describes a two-component Job that updates data in a MySQL table according to that in a delimited file.

- Drop tFileInputDelimited and tMysqlOutput from the Palette onto the design workspace.
- Connect the two components together using a Row Main link.

- Double-click tFileInputDelimited to display its Basic settings view and define the component properties.
- From the Property Type list, select Repository if you have already stored the metadata of the delimited file in the Metadata node in the Repository tree view. Otherwise, select Built-In to define manually the metadata of the delimited file.

For more information about storing metadata, see Talend Studio User Guide.
• In the **File Name** field, click the three-dot button and browse to the source delimited file that contains the modifications to propagate in the MySQL table.

In this example, we use the *customer_update* file that holds four columns: `id`, `CustomerName`, `CustomerAddress` and `idState`. Some of the data in these four columns is different from that in the MySQL table.

<table>
<thead>
<tr>
<th>id</th>
<th>CustomerName</th>
<th>CustomerAddress</th>
<th>idState</th>
</tr>
</thead>
<tbody>
<tr>
<td>858</td>
<td>Froggy's Gourmet Catering</td>
<td>1831 Beverly #9D</td>
<td>4</td>
</tr>
<tr>
<td>859</td>
<td>Dependable Plumbing and Sewer</td>
<td>1550 Ridge Rd.</td>
<td>25</td>
</tr>
<tr>
<td>860</td>
<td>Lockman Restoration</td>
<td>1235 Easton Rd.</td>
<td>40</td>
</tr>
<tr>
<td>861</td>
<td>Actuarial Enterprises Ltd.</td>
<td>3148 cottonwood Ct.</td>
<td>18</td>
</tr>
<tr>
<td>862</td>
<td>Rythmics Ltd.</td>
<td>857 Woodbine Rd.</td>
<td>30</td>
</tr>
<tr>
<td>863</td>
<td>Actuarial Enterprises Ltd.</td>
<td>1482 Concorde Circle</td>
<td>48</td>
</tr>
<tr>
<td>864</td>
<td>Crosstracks Car Wash</td>
<td>218 Oakridge Ave.</td>
<td>39</td>
</tr>
<tr>
<td>865</td>
<td>Meonis &amp; Mogoni Inc.</td>
<td>616 Cobblestone Cir.</td>
<td>17</td>
</tr>
<tr>
<td>866</td>
<td>Foley Aviation</td>
<td>2220 Grant Blvd.</td>
<td>50</td>
</tr>
<tr>
<td>867</td>
<td>Ebert Music Center</td>
<td>12 Broadview Lane</td>
<td>29</td>
</tr>
<tr>
<td>868</td>
<td>Janice Mann Accounting Service</td>
<td>1660 Park Ave.</td>
<td>9</td>
</tr>
<tr>
<td>869</td>
<td>Johnson, Erico &amp; Co CPA's</td>
<td>2922 Twin Oaks Drive</td>
<td>40</td>
</tr>
<tr>
<td>870</td>
<td>Corbins, Rodriguez, &amp; Savocchi</td>
<td>115 Pleasant Ave.</td>
<td>18</td>
</tr>
<tr>
<td>871</td>
<td>Nina's Snow Plowing</td>
<td>3385 University Ave.</td>
<td>20</td>
</tr>
<tr>
<td>872</td>
<td>Darcy Frame and Matting Servic</td>
<td>1101 Deffield Place</td>
<td>47</td>
</tr>
<tr>
<td>873</td>
<td>Marks, Marks, and Kaplan Ltd.</td>
<td>1949 Cleaverdale Rd.</td>
<td>9</td>
</tr>
</tbody>
</table>

• Define the row and field separators used in the source file in the corresponding fields.

• If needed, set **Header**, **Footer** and **Limit**.

In this example, **Header** is set to 1 since the first row holds the names of columns, therefore it should be ignored. Also, the number of processed lines is limited to 2000.

• Click the three-dot button next to **Edit Schema** to open a dialog box where you can describe the data structure of the source delimited file that you want to pass to the component that follows.

<table>
<thead>
<tr>
<th>Column</th>
<th>Key</th>
<th>Type</th>
<th>Nullable</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>✓</td>
<td>Integer</td>
<td>✓</td>
</tr>
<tr>
<td>CustomerName</td>
<td></td>
<td>String</td>
<td>✓</td>
</tr>
<tr>
<td>CustomerAddress</td>
<td></td>
<td>String</td>
<td>✓</td>
</tr>
<tr>
<td>idState</td>
<td></td>
<td>Integer</td>
<td>✓</td>
</tr>
</tbody>
</table>

• Select the **Key** check box(es) next to the column name(s) you want to define as key column(s).

**Note:**

It is necessary to define at least one column as a key column for the Job to be executed correctly. Otherwise, the Job is automatically interrupted and an error message displays on the console.

• In the design workspace, double-click **tMysqlOutput** to open its **Basic settings** view where you can define its properties.
Click **Sync columns** to retrieve the schema of the preceding component. If needed, click the three-dot button next to **Edit schema** to open a dialog box where you can check the retrieved schema.

From the **Property Type** list, select **Repository** if you have already stored the connection metadata in the **Metadata** node in the **Repository** tree view. Otherwise, select **Built-In** to define manually the connection information.

For more information about storing metadata, see *Talend Studio User Guide*.

Fill in the database connection information in the corresponding fields.

In the **Table** field, enter the name of the table to update.

From the **Action on table** list, select the operation you want to perform, **Default** in this example since the table already exists.

From the **Action on data** list, select the operation you want to perform on the data, **Update** in this example.

Save your Job and press **F6** to execute it.
Using your DB browser, you can verify if the MySQL table, customers, has been modified according to the delimited file.

In the above example, the database table has always the four columns id, CustomerName, CustomerAddress and idState, but certain fields have been modified according to the data in the delimited file used.

**Scenario: Retrieving data in error with a Reject link**

This scenario describes a four-component Job that carries out migration from a customer file to a MySQL database table and redirects data in error towards a CSV file using a Reject link.
• In the Repository, select the customer file metadata that you want to migrate and drop it onto the workspace. In the [Components] dialog box, select tFileInputDelimited and click OK. The component properties will be filled in automatically.

• If you have not stored the information about your customer file under the Metadata node in the Repository, drop a tFileInputDelimited component from the family File > Input, in the Palette, and fill in its properties manually in the Component tab.

• From the Palette, drop a tMap from the Processing family onto the workspace.

• In the Repository, expand the Metadata node, followed by the Db Connections node and select the connection required to migrate your data to the appropriate database. Drop it onto the workspace. In the [Components] dialog box, select tMysqlOutput and click OK. The database connection properties will be automatically filled in.

• If you have not stored the database connection details under the Db Connections node in the Repository, drop a tMysqlOutput from the Databases family in the Palette and fill in its properties manually in the Component tab.

For more information, see Talend Studio User Guide.

• From the Palette, select a tFileOutputDelimited from the File > Output family, and drop it onto the workspace.

• Link the customers component to the tMap component, and the tMap and Localhost with a Row Main link. Name this second link out.

• Link the Localhost to the tFileOutputDelimited using a Row > Reject link.

• Double-click the customers component to display the Component view.

• In the Property Type list, select Repository and click the [...] button in order to select the metadata containing the connection to your file. You can also select the Built-in mode and fill in the fields manually.

• Click the [...] button next to the File Name field, and fill in the path and the name of the file you want to use.

• In the Row and Field Separator fields, type in between inverted commas the row and field separator used in the file.

• In the Header, Footer and Limit fields, type in the number of headers and footers to ignore, and the number of rows to which processing should be limited.
In the **Schema** list, select **Repository** and click the [...] button in order to select the schema of your file, if it is stored under the **Metadata** node in the **Repository**. You can also click the [...] button next to the **Edit schema** field, and set the schema manually.

The schema is as follows:

![Schema of customers](image)

- Double-click the **tMap** component to open its editor.

![tMap component](image)

- Select the `id`, `CustomerName`, `CustomerAddress`, `idState`, `id2`, `RegTime` and `RegisterTime` columns on the table on the left and drop them on the **out** table, on the right.
• In the **Schema editor** area, at the bottom of the **tMap** editor, in the right table, change the length of the **CustomerName** column to 28 to create an error. Thus, any data for which the length is greater than 28 will create errors, retrieved with the **Reject** link.

• Click **OK**.

• In the workspace, double-click the output **Localhost** component to display its **Component** view.

• In the **Property Type** list, select **Repository** and click the [...] button to select the connection to the database metadata. The connection details will be automatically filled in. You can also select the **Built-in** mode and set the fields manually.

• In the **Table** field, type in the name of the table to be created. In this scenario, we call it **customers_data**.

• In the **Action on data** list, select the **Create table** option.

• Click the **Sync columns** button to retrieve the schema from the previous component.

• Make sure the **Die on error** check box isn’t selected, so that the Job can be executed despite the error you just created.
• Click the **Advanced settings** tab of the **Component** view to set the advanced parameters of the component.

![mysqloutput](image)

- Deselect the **Extend Insert** check box which enables you to insert rows in batch, because this option is not compatible with the **Reject** link.

- Double-click the **tFileOutputDelimited** component to set its properties in the **Component** view.

![fileoutputdelimited](image)

• Click the [...] button next to the **File Name** field to fill in the path and name of the output file.

• Click the **Sync columns** button to retrieve the schema of the previous component.

• Save your Job and press **F6** to execute it.
<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>ID</th>
<th>Event Time</th>
<th>Error Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Griffith Pav-(tand@apres</td>
<td>7</td>
<td>41</td>
<td>2001-01-17 06:26</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Terrini &amp; Sol770</td>
<td>5</td>
<td>9</td>
<td>1982-04-16 10:26</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Elle Hypnosis</td>
<td>1</td>
<td>7</td>
<td>1975-06-10 20:20</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Darcy Frame</td>
<td>21</td>
<td>20</td>
<td>2001-01-07 20:40</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Salt &amp; Peppe</td>
<td>44</td>
<td>33</td>
<td>2006-01-24 10:53</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Arthur M. Fei</td>
<td>37</td>
<td>24</td>
<td>1974-07-24 20:40</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Got It All Elet629</td>
<td>22</td>
<td>28</td>
<td>1993-10-22 15:26</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Crossroads C2267</td>
<td>5</td>
<td>14</td>
<td>1977-10-24 17:53</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Jay's Cabinet</td>
<td>48</td>
<td>43</td>
<td>2006-02-29 07:46</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Lee Stanwork</td>
<td>41</td>
<td>46</td>
<td>1992-04-15 09:41</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Lambert &amp; Lo662</td>
<td>45</td>
<td>17</td>
<td>2002-11-22 04:26</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Janice Mann</td>
<td>28</td>
<td>35</td>
<td>1983-04-21 15:26</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Michael Mont</td>
<td>9</td>
<td>9</td>
<td>2000-10-07 03:26</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Dean Barnett</td>
<td>18</td>
<td>46</td>
<td>2006-03-23 10:14</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Dooner's Aw618</td>
<td>26</td>
<td>28</td>
<td>1993-05-03 04:33</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Auto Interior</td>
<td>48</td>
<td>40</td>
<td>2006-04-25 13:14</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>First National</td>
<td>5</td>
<td>41</td>
<td>2003-09-23 01:54</td>
<td>Data truncation: Data too long</td>
</tr>
<tr>
<td>Terrini &amp; Sol</td>
<td>12</td>
<td>44</td>
<td>1996-12-26 05:46</td>
<td>Data truncation: Data too long</td>
</tr>
</tbody>
</table>

The data in error are sent to the delimited file, as well as the error type met. Here, we have: Data truncation.
tMysqlOutputBulk

Writes a file with columns based on the defined delimiter and the MySQL standards.

The tMysqlOutputBulk and tMysqlBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT statement used to feed a database. These two steps are fused together in the tMysqlOutputBulkExec component, detailed in a separate section. The advantage of using two separate steps is that the data can be transformed before it is loaded in the database.

**tMysqlOutputBulk Standard properties**

These properties are used to configure tMysqlOutputBulk running in the Standard Job framework.

The Standard tMysqlOutputBulk component belongs to the Databases family.

The component in this framework is generally available.

Note: This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>File Name</td>
<td>Name of the file to be generated. This file is generated on the same machine where the Studio is installed or where the Job using tMysqlOutputBulk is deployed.</td>
</tr>
<tr>
<td>Append</td>
<td>Select this check box to add the new rows at the end of the file.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>
Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

### Advanced settings

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row separator</strong></td>
<td>String (ex: “\n” on Unix) to distinguish rows.</td>
</tr>
<tr>
<td><strong>Field separator</strong></td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td><strong>Text enclosure</strong></td>
<td>Character used to enclose the text.</td>
</tr>
<tr>
<td><strong>Create directory if does not exist</strong></td>
<td>This check box is selected by default. It creates a directory to hold the output table if required.</td>
</tr>
<tr>
<td><strong>Custom the flush buffer size</strong></td>
<td>Customize the amount of memory used to temporarily store output data. In the Row number field, enter the number of rows after which the memory is to be freed again.</td>
</tr>
<tr>
<td><strong>Records contain NULL value</strong></td>
<td>This check box is selected by default. It allows you to take account of NULL value fields. If you clear the check box, the NULL values will automatically be replaced with empty values.</td>
</tr>
<tr>
<td><strong>Check disk space</strong></td>
<td>Select the this check box to throw an exception during execution if the disk is full.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select the encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling.</td>
</tr>
</tbody>
</table>
### Global Variables

**Global Variables**

**NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.


### Usage

**Usage rule**

This component is to be used along with **tMySQlBulkExec** component. Used together they offer gains in performance while feeding a MySQL database.

**Component family**

Databases/MySQL

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

### Scenario: Inserting transformed data in MySQL database

This scenario describes a four-component job which aims at fueling a database with data contained in a file, including transformed data. Two steps are required in this job, first step is to create the file, that will then be used in the second step. The first step includes a tranformation phase of the data included in the file.
Dropping and linking components

Procedure
1. Drag and drop a tRowGenerator, a tMap, a tMysqlOutputBulk as well as a tMysqlBulkExec component.
2. Connect the main flow using row Main links.
3. And connect the start component (tRowGenerator in this example) to the tMysqlBulkExec using a trigger connection, of type OnComponentOk.

Configuring the components

Procedure
1. A tRowGenerator is used to generate random data. Double-click on the tRowGenerator component to launch the editor.
2. Define the schema of the rows to be generated and the nature of data to generate. In this example, the clients file to be produced will contain the following columns: ID, First Name, Last Name, Address, City which all are defined as string data but the ID that is of integer type.
Some schema information don’t necessarily need to be displayed. To hide them away, click on **Columns** list button next to the toolbar, and uncheck the relevant entries, such as **Precision** or **Parameters**.

Use the plus button to add as many columns to your schema definition. Click the Refresh button to preview the first generated row of your output.

3. Then select the **tMap** component to set the transformation.
4. Drag and drop all columns from the input table to the output table.

5. Apply the transformation on the **LastName** column by adding `.toUpperCase()` in its expression field. Then, click **OK** to validate the transformation.
6. Double-click on the **tMysqlOutputBulk** component.
7. Define the name of the file to be produced in **File Name** field. If the delimited file information is stored in the **Repository**, select it in **Property Type** field, to retrieve relevant data. In this use case the file name is **clients.txt**.
   
   The schema is propagated from the **tMap** component, if you accepted it when prompted.
8. In this example, don’t include the header information as the table should already contain it.
9. Click OK to validate the output.
10. Then double-click on the **tMysqlBulkExec** component to set the INSERT query to be executed.
11. Define the database connection details. We recommend you to store this type of information in the Repository, so that you can retrieve them at any time for any Job.

12. Set the table to be filled in with the collected data, in the Table field.

13. Fill in the column delimiters in the Field terminated by area.

14. Make sure the encoding corresponds to the data encoding.

Saving and executing the Job

Procedure

1. Press Ctrl+S to save your Job.

2. Press F6 or click Run on the Run tab to execute the Job.

The clients database table is filled with data from the file including upper-case last name as transformed in the job.

Results

For simple Insert operations that don’t include any transformations, the use of tMysqlOutputBulkExec allows you to skip a step in the process and thus improves performance.

Related topic: tMysqlOutputBulkExec Standard properties on page 2541
tMysqlOutputBulkExec

Executes the Insert action in the specified MySQL database.

As a dedicated component, tMysqlBulkExec improves performance during Insert operations to a MySQL database.

The tMysqlOutputBulk and tMysqlBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT statement used to feed a database. These two steps are fused together in the tMysqlOutputBulkExec component.

tMysqlOutputBulkExec Standard properties

These properties are used to configure tMysqlOutputBulkExec running in the Standard Job framework. The Standard tMysqlOutputBulkExec component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the</td>
</tr>
<tr>
<td></td>
<td>properties are stored. The fields that follow are</td>
</tr>
<tr>
<td></td>
<td>completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the version of MySQL that you are using.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to</td>
</tr>
<tr>
<td></td>
<td>the password field, and then in the pop-up dialog box</td>
</tr>
<tr>
<td></td>
<td>enter the password between double quotes and click OK to</td>
</tr>
<tr>
<td></td>
<td>save the settings.</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
</tbody>
</table>
## None: No operation is carried out.

**Drop and create table:** The table is removed and created again.

**Create table:** The table does not exist and gets created.

**Create table if not exists:** The table is created if it does not already exist.

**Drop table if exists and create:** The table is removed if it already exists and created again.

**Clear table:** The table content is deleted.

### Table

Name of the table to be written.

**Note:**

Note that only one table can be written at a time and that the table must already exist for the insert operation to succeed.

### Local FileName

Name of the file to be generated and loaded.

This file is generated on the same machine where the Studio is installed or where the Job using tMysqlOutputBulkExec is deployed. Then it is loaded to the database specified by Host field.

### Append

Select the check box for this option to append new rows to the end of the file.

### Schema and Edit Schema

A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-In:** You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

**Repository:** You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.
When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Additional JDBC parameters</strong></th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <strong>Use an existing connection</strong> check box in the Basic settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>You can press <strong>Ctrl+Space</strong> to access a list of predefined global variables.</td>
</tr>
<tr>
<td><strong>Row separator</strong></td>
<td>String (ex: <code>\n</code> on Unix) to distinguish rows.</td>
</tr>
<tr>
<td><strong>Field separator</strong></td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td><strong>Escape char</strong></td>
<td>Character of the row to be escaped</td>
</tr>
<tr>
<td><strong>Text enclosure</strong></td>
<td>Character used to enclose the text.</td>
</tr>
<tr>
<td><strong>Create directory if does not exist</strong></td>
<td>This check box is selected by default. It creates a directory to hold the output table if required.</td>
</tr>
<tr>
<td><strong>Custom the flush buffer size</strong></td>
<td>Customize the amount of memory used to temporarily store output data. In the <strong>Row number field</strong>, enter the number of rows after which the memory is to be freed again.</td>
</tr>
<tr>
<td><strong>Action on data</strong></td>
<td>On the data of the table defined, you can carry out the following operations:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert records in table:</strong> Add new records to the table.</td>
</tr>
<tr>
<td></td>
<td><strong>Update records in table:</strong> Make changes to existing records.</td>
</tr>
<tr>
<td></td>
<td><strong>Replace records in table:</strong> Replace existing records with new one. <strong>Ignore records in table:</strong> Ignore existing records or insert the new ones.</td>
</tr>
<tr>
<td><strong>Records contain NULL value</strong></td>
<td>This check box is selected by default. It allows you to take account of NULL value fields. If you clear the check box, the NULL values will automatically be replaced with empty values.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect the log data at the component level.</td>
</tr>
</tbody>
</table>
Usage

Usage rule

This component is mainly used when no particular transformation is required on the data to be loaded onto the database.

Scenario: Inserting data in bulk in MySQL database

This scenario describes a two-component Job which carries out the same operation as the one described for tMysqlOutputBulk Standard properties on page 2535 and tMysqlBulkExec Standard properties on page 2461, although no data is transformed.

- Drop a tRowGenerator and a tMysqlOutputBulkExec component from the Palette to the design workspace.
- Connect the components using a link such as Row > Main.
- Set the tRowGenerator parameters the same way as in Scenario: Inserting transformed data in MySQL database on page 2537. The schema is made of four columns including: ID, First Name, Last Name, Address and City.
- In the workspace, double-click the tMysqlOutputBulkExec to display the Component view and set the properties.

- In the Action on table field, select the None option as you want to insert the data into a table which already exists.

For further information, see Talend Studio User Guide.
In the **Table** field, enter the name of the table you want to populate, the name being *clients* in this example.

In the **Local filename** field, indicate the access path and the name of the file which contains the data to be added to the table. In this example, the file is *clients.txt*.

Click on the **Advanced settings** tab to define the component’s advanced parameters.

- In the **Action on data** list, select the **Insert records in table** to insert the new data in the table.

Press **F6** to run the Job.

The result should be pretty much the same as in **Scenario: Inserting transformed data in MySQL database** on page 2537, but the data might differ as these are regenerated randomly everytime the Job is run.
tMysqlRollback

Cancels the transaction commit in the connected MySQL database to avoid committing part of a transaction involuntarily.

**tMysqlRollback Standard properties**

These properties are used to configure tMysqlRollback running in the Standard Job framework.

The Standard tMysqlRollback component belongs to the Databases family.

The component in this framework is generally available.

**i Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tMysqlConnection</strong> component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

### Advanced settings

| tStat Catcher Statistics                         | Select this check box to collect log data at the component level. |

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other tMysql* components, especially with the <strong>tMysqlConnection</strong> and <strong>tMysqlCommit</strong> components.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the <code>[+]</code> button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</td>
</tr>
</tbody>
</table>
Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.
tMysqlRow

Executes the stated SQL query on the specified MySQL database.

Depending on the nature of the query and the database, tMysqlRow acts on the actual DB structure or on the data (although without handling data). The SQLBuilder tool helps you write easily your SQL statements.

tMysqlRow is the specific component for MySQL database query. It implements a flow in the Job design although it doesn’t provide output.

### tMysqlRow Standard properties

These properties are used to configure tMysqlRow running in the Standard Job framework.

The Standard tMysqlRow component belongs to the Databases family.

The component in this framework is generally available.

#### Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

#### Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>DB Version</strong></td>
<td>Select the MySQL version that you are using.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

#### Note:

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see **Talend Studio User Guide**.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <strong>Talend Studio User Guide</strong>. When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
<td></td>
</tr>
</tbody>
</table>

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Name of the table to be processed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: Fill in manually the query statement or build it graphically using SQLBuilder.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</td>
</tr>
<tr>
<td>Guess Query</td>
<td>Click the Guess Query button to generate the query which corresponds to your table schema in the Query field.</td>
</tr>
<tr>
<td>Query</td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row &gt; Rejects link.</td>
</tr>
<tr>
<td>Specify a data source alias</td>
<td>Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503. If you use the component's own DB configuration, your data source connection will be closed at the end of the component. To prevent this from happening, use a shared DB connection with the data source alias specified. This check box is not available when the Use an existing connection check box is selected.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| Additional JDBC parameters       | Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the Use an existing connection check box in the Basic settings. |
| Propagate QUERY’s recordset     | Select this check box to insert the result of the query in a COLUMN of the current flow. Select this column from the use column list.  

**Note:**  
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the
type of Object and this component is usually followed by tParseRecordSet.

Use PreparedStatement

Select this check box if you want to query the database using a PreparedStatement. In the Set PreparedStatement Parameter table, define the parameters represented by "?" in the SQL instruction of the Query field in the Basic Settings tab.

Parameter Index: Enter the parameter position in the SQL instruction.
Parameter Type: Enter the parameter type.
Parameter Value: Enter the parameter value.

Note:
This option is very useful if you need to execute the same query several times. Performance levels are increased.

Commit every

Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.

tStatCatcher Statistics

Select this check box to collect log data at the component level.

Global Variables

QUERY: the query statement being processed. This is a Flow variable and it returns a string.

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

This component offers the flexibility of the DB query and covers all possible SQL queries.

Dynamic settings

Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful.
when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Scenario: Removing and regenerating a MySQL table index

This scenario describes a four-component job that removes a table index, applies a select insert action onto a table then regenerates the index.

#### Linking the components

**Procedure**

1. Select and drop the following components onto the design workspace: **tMysqlRow** (x2), **tRowGenerator**, and **tMysqlOutput**.
2. Link the first **tMysqlRow** to **tRowGenerator** using an OnComponentOk connection.
3. Link **tRowGenerator** to **tMysqlOutput** using a Row > Main connection.
4. Link **tRowGenerator** to the second **tMysqlRow** using an OnSubjobOk connection.
Configuring the components

Procedure

1. Select the tMysqlRow to fill in the DB Basic settings.
2. In Property type as well in Schema, select the relevant DB entry in the list.
   The DB connection details and the table schema are accordingly filled in.
3. Propagate the properties and schema details onto the other components of the Job.
   The query being stored in the Metadata area of the Repository, you can also select Repository in the Query type field and the relevant query entry.
4. If you didn’t store your query in the Repository, type in the following SQL statement to alter the database entries: drop index <index_name> on <table_name>
5. Select the second tMysqlRow component, check the DB properties and schema.
6. Type in the SQL statement to recreate an index on the table using the following statement: create index <index_name> on <table_name> (<column_name>)
   The tRowGenerator component is used to generate automatically the columns to be added to the DB output table defined.
7. Select the tMysqlOutput component and fill in the DB connection properties either from the Repository or manually the DB connection details are specific for this use only. The table to be fed is named: comprehensive.
8. The schema should be automatically inherited from the data flow coming from the tLogRow. Edit the schema to check its structure and check that it corresponds to the schema expected on the DB table specified.
   The Action on table is None and the Action on data is Insert.
   No additional Columns is required for this job.

Executing the Job

Procedure

1. Press Ctrl+S to save the Job.
2. Press F6 to run the job.
   If you manage to watch the action on DB data, you can notice that the index is dropped at the start of the job and recreated at the end of the insert action.
   Related topics: tDBSQLRow Standard properties on page 622.

Scenario: Using PreparedStatement objects to query data

This scenario describes a four component job which allows you to link a table column with a client file. The MySQL table contains a list of all the American States along with the State ID, while the file contains the customer information including the ID of the State in which they live. We want to retrieve the name of the State for each client, using an SQL query. In order to process a large volume of data quickly, we use a PreparedStatement object which means that the query is executed only once rather than against each row in turn. Then each row is sent as a parameter. Note that PreparedStatement object can also be used in avoiding SQL injection.

For this scenario, we use a file and a database for which we have already stored the connection and properties in the Repository metadata. For further information concerning the creation of metadata in
delimited files, the creation of database connection metadata and the usage of metadata, see *Talend Studio User Guide*.

### Linking the components

**Procedure**

1. In the **Repository**, expand the **Metadata** and **File delimited** nodes.
2. Select the metadata which corresponds to the client file and slide the metadata onto the workspace. Here, we are using the `customers` metadata.
3. Double-click **tFileInputDelimited** in the **Components** dialog box to add **tFileInputDelimited** to the workspace, with the relevant fields filled by the metadata file.
4. Drop **tMysqlRow**, **tParseRecordSet** and **tFileOutputDelimited** onto the workspace.
5. Link **tFileInputDelimited** to **tMysqlRow** using a **Row > Main** connection.
6. Link **tMysqlRow** to **tParseRecordSet** using a **Row > Main** connection.
7. Link **tParseRecordSet** to **tFileOutputDelimited** using a **Row > Main** connection.

### Configuring the components

**Procedure**

1. Double-click **tFileInputDelimited** to open its **Basic settings** view.

   ![Basic settings view](image)

   **Basic settings**
   - **Property/Type**: Repository, DELIM:customers
   - **File name/Stream**: `D:/04_cobs/Files/customer.csv`
   - **Row Separator**: `|`
   - **Field Separator**: `;`
   - **Header**: 6
   - **Footer**: 0
   - **Limit**: 0
   - **Schema**: Repository, DELIM:customers - metadata

2. In the **Schema** list, select **Built-in** so that you can modify the component's schema. Then click on **[...]** next to the **Edit schema** field to add a column into which the name of the State will be inserted.
3. Click on the [+] button to add a column to the schema. Rename this column `LabelStateRecordSet` and select `Object` from the `Type` list. Click `OK` to save your modifications.

4. From the Palette, select the `tMysqlRow`, `tParseRecordSet` and `tFileOutputDelimited` components and drop them onto the workspace.

5. Double click `tMysqlRow` to set its properties in the `Basic settings` tab of the `Component` view.
6. In the **Property Type** list, select **Repository** and click on the [...] button to select a database connection from the metadata in the Repository. The **DB Version**, **Host**, **Port**, **Database**, **Username** and **Password** fields are completed automatically. If you are using the **Built-in** mode, complete these fields manually.

7. From the **Schema** list, select **Built-in** to set the schema properties manually and add the **LabelStateRecordSet** column, or click directly on the **Sync columns** button to retrieve the schema from the preceding component.

8. In the **Query** field, enter the SQL query you want to use. Here, we want to retrieve the names of the American States from the **LabelState** column of the MySQL table, **us_state**: "SELECT LabelState FROM us_state WHERE idState=?". The question mark, "?", represents the parameter to be set in the **Advanced settings** tab.

9. Click **Advanced settings** to set the components advanced properties.

10. Select the **Propagate QUERY's recordset** check box and select the **LabelStateRecordSet** column from the **use column** list to insert the query results in that column.

11. Select the **Use PreparedStatement** check box and define the parameter used in the query in the **Set PreparedStatement Parameters** table.

12. Click on the [+] button to add a parameter.

   a) In the **Parameter Index** cell, enter the parameter position in the SQL instruction. Enter "1" as we are only using one parameter in this example.

   b) In the **Parameter Type** cell, enter the type of parameter. Here, the parameter is a whole number, hence, select **Int** from the list.

   c) In the **Parameter Value** cell, enter the parameter value. Here, we want to retrieve the name of the State based on the State ID for every client in the input file. Hence, enter "row1.idState".

13. Double click **tParseRecordSet** to set its properties in the **Basic settings** tab of the **Component** view.
a) From the **Prev. Comp. Column list**, select the preceding components column for analysis. In this example, select *LabelStateRecordSet*.

b) Click on the **Sync columns** button to retrieve the schema from the preceding component. The **Attribute table** is automatically completed with the schema columns.

c) In the **Attribute table**, in the **Value** field which corresponds to the *LabelStateRecordSet*, enter the name of the column containing the State names to be retrieved and matched with each client, within double quotation marks. In this example, enter "*LabelState*".

14. Double click **tFileOutputDelimited** to set its properties in the **Basic settings** tab of the **Component** view.

a) In the **File Name field**, enter the access path and name of the output file.

b) Click **Sync columns** to enter the schema from the preceding component.

### Executing the Job

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to run it.
Results

A column containing the name of the American State corresponding to each client is added to the file.

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Griffith</td>
<td>Paving</td>
<td>Island</td>
<td>Apres</td>
<td>7</td>
<td>41</td>
<td>03/11/2006</td>
<td>09:20</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Childress</td>
<td>Chi</td>
<td>662</td>
<td>Lyons</td>
<td>Ci</td>
<td>1</td>
<td>28</td>
<td>16/02/2005</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Terrini</td>
<td>&amp;</td>
<td>Stol</td>
<td>770</td>
<td>Exmoor</td>
<td>F</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>Tub's</td>
<td>Furnitur</td>
<td>607</td>
<td>Old</td>
<td>Trail</td>
<td>15</td>
<td>9</td>
<td>27/08/2000</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>Toggle</td>
<td>&amp;</td>
<td>Mye</td>
<td>618</td>
<td>Sheridan</td>
<td>9</td>
<td>15</td>
<td>24/03/2006</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>Childress</td>
<td>Chi</td>
<td>708</td>
<td>Tennis</td>
<td>12</td>
<td>33</td>
<td>10/09/2001</td>
<td>06:33</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>Elle</td>
<td>Hypnosis</td>
<td>2032</td>
<td>North</td>
<td>1</td>
<td>7</td>
<td>11/01/1977</td>
<td>03:07</td>
</tr>
</tbody>
</table>

Scenario: Combining two flows for selective output

In this scenario, a flow generated by tFixedFlowInput is combined with a flow from the Mysql database. The source flow contains id and age fields while the Mysql table contains id and name. We want to retrieve the age data of the source flow and combine it with the id and name records from the Mysql table based on id matching. One thing that is worth noting is that the input schema is different from the output one at tMysqlRow in the Job.

Linking the components

Procedure

1. Drop tFixedFlowInput, tMysqlRow, tParseRecordSet and tLogRow onto the workspace.
2. Rename tFixedFlowInput as source_flow, tMysqlRow as insert_recordset, tParseRecordSet as parse_recordset and tLogRow as show_combined_flow.
3. Link tFixedFlowInput to tMysqlRow using a Row > Main connection.
4. Link tMysqlRow to tParseRecordSet using a Row > Main connection.
5. Link tParseRecordSet to tLogRow using a Row > Main connection.

Configuring the components

Procedure

1. Double-click tFixedFlowInput to open its Basic settings view.
2. Select **Use Inline Content (delimited file)** in the **Mode** area. In the **Content** field, enter the data to be transferred:

```
1;30
2;20
```

3. Double-click the [...] button next to **Edit schema** to open the schema editor.

Click the [+] button to add two columns, namely *id* and *age*, with the type of *Integer*. Click **Ok** to close the editor.

4. Double-click **tMysqlRow** to open its **Basic settings** view.
5. In the Host and Port fields, enter the connection details.
   In the Database field, enter the database name.
   In the Username and Password fields, enter the authentication details.
   In the Query field, enter the SQL query to retrieve the id and name data from the Mysql table employee: "select id, name from employee WHERE id=?". The question mark, ",\?", represents the parameter to be set in the Advanced settings tab.

6. Click the [...] button next to Edit schema to open the schema editor.

7. Click the [+] button to add two columns in the right part, namely recordset and age, with the type of Object and Integer. Note that recordset is intended to hold the query results of the Mysql table, namely the id and name fields.
   Click OK to close the editor.

8. Click the Advanced settings tab for further setup.
9. Select the Propagate QUERY's recordset check box and choose recordset from the use column list to insert the query results in that column.

Select the Use PreparedStatement check box and define the parameter used in the query in the Set PreparedStatement Parameters table.

10. Click on the [+] button to add a line.

In the Parameter Index cell, enter the parameter position in the SQL instruction. Enter "1" as we are only using one parameter in this example.

In the Parameter Type cell, enter the type of parameter. Here, the parameter is an integer. Hence, select Int from the list.

In the Parameter Value cell, enter the parameter value. Here, we want to retrieve the id and name from the employee table based on the id value from the source flow. Hence, enter row3.id.

11. Double-click tParseRecordSet to open its Basic settings view.

From the Prev. Comp. Column list, select the column to parse, namely recordset.

12. Click the [...] button next to Edit schema to open the schema editor.
Click the [+] button to add three columns in the right part, namely *id*, *name* and *age*, with the type of *Integer*, *String* and *Integer*. Note that the *id* and *name* fields are intended to hold the parsed data of *recordset*.

Click OK to close the editor.

In the **Attribute table**, in the **Value** fields which correspond to *id* and *name*, enter the name of the column in the Mysql table to be retrieved, namely “*id*” and “*name*”.

13. Double-click **tLogRow** to open its **Basic settings** view.

In the **Mode** area, select **Table (print values in cells of a table)** for better display.

**Executing the Job**

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to run the Job.

```
[statistics] connecting to socket on port 4030
[statistics] connected
-----------
<table>
<thead>
<tr>
<th>show_combined_flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>
-----------
[statistics] disconnected
```
tMysqlSCD

Reflects and tracks changes in a dedicated MySQL SCD table.
tMysqlSCD addresses Slowly Changing Dimension needs, reading regularly a source of data and logging the changes into a dedicated SCD table.

**tMysqlSCD Standard properties**

These properties are used to configure tMysqlSCD running in the Standard Job framework.
The Standard tMysqlSCD component belongs to the Business Intelligence and the Databases families.
The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the Repository file where properties are stored. The following fields are pre-filled in using fetched data.</td>
<td></td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

| **DB Version** | Select the Mysql version you are using. |
| **Host** | Database server IP address. |
| **Port** | Listening port number of DB server. |
| **Database** | Name of the database. |
| **Username and Password** | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Table** | Name of the table to be written. Note that only one table can be written at a time. |
| **Action on table** | On the table defined, you can perform one of the following operations:  
| None: No operation is carried out.  
| Create a table: The table does not exist and gets created.  
| Create a table if not exists: The table is created if it does not exist. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
| Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
| • View schema: choose this option to view the schema only.  
| • Change to built-in property: choose this option to change the schema to Built-in for local changes.  
| • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.  
| Built-in: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.  
<p>| Repository: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <em>Talend Studio User Guide</em>. |</p>
<table>
<thead>
<tr>
<th><strong>SCD Editor</strong></th>
<th>The SCD editor helps to build and configure the data flow for slowly changing dimension outputs. For more information, see SCD management methodology on page 2566.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use memory saving mode</strong></td>
<td>Select this check box to maximize system performance.</td>
</tr>
</tbody>
</table>
| **Source keys include Null** | Select this check box to allow the source key columns to have Null values.  

⚠️ **Warning:**  
Special attention should be paid to the uniqueness of the source key(s) values when this option is selected. |
| **Die on error** | This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows. |

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Additional JDBC Parameters</strong></th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <strong>Use an existing connection</strong> check box in the Basic settings.</th>
</tr>
</thead>
</table>
| **End date time details** | Specify the time value of the SCD end date time setting in the format of **HH:mm:ss**. The default value for this field is **12:00:00**.  
This field appears only when SCD Type 2 is used and Fixed year value is selected for creating the SCD end date. |
| **tStatCatcher Statistics** | Select this check box to collect log data at the component level. |
| **Debug mode** | Select this check box to display each step during processing entries in a database. |

### Global Variables

| **Global Variables** | **NB_LINE_UPDATED**: the number of rows updated. This is an After variable and it returns an integer.  
**NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.  
**NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
# SCD management methodology

Slowly Changing Dimensions (SCDs) are dimensions that change slowly over time.

The SCD editor offers the simplest method of building the data flow for the SCD outputs. In the SCD editor, you can map columns, select surrogate key columns, and set column change attributes through combining SCD types. The following figure illustrates an example of the SCD editor.

## Usage

### Usage rule

This component is used as Output component. It requires an Input component and Row main link as input.

### Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Limitation

This component does not support using SCD type 0 together with other SCD types.
**SCD keys**

You must choose one or more source keys columns from the incoming data to ensure its unicity.

You must set one surrogate key column in the dimension table and map it to an input column in the source table. The value of the surrogate key links a record in the source to a record in the dimension table. The editor uses this mapping to locate the record in the dimension table and to determine whether a record is new or changing. The surrogate key is typically the primary key in the source, but it can be an alternate key as long as it uniquely identifies a record and its value does not change.

**Source keys**: Drag one or more columns from the **Unused** panel to the **Source keys** panel to be used as the key(s) that ensure the unicity of the incoming data.

**Surrogate keys**: Set the column where the generated surrogate key will be stored. A surrogate key can be generated based on a method selected on the **Creation** list.

**Creation**: Select any of the below methods to be used for the key generation.

- **Auto increment**: auto-incremental key.
- **Input field**: key is provided in an input field.
When selected, you can drag the appropriate field from the **Unused** panel to the **complement** field.

- **Routine**: from the **complement** field, you can press **Ctrl+Space** to display the autocompletion list and select the appropriate routine.

- **Table max +1**: the maximum value from the SCD table is incremented to create a surrogate key.

- **DB Sequence**: from the **complement** field, you can enter the name of the existing database sequence that will automatically increment the column indicated in the **name** field.

Note that this option is only available through the **SCD Editor** of the **tOracleSCD**.

### Slowly Changing Dimensions types

The Slowly Changing Dimensions support four types of changes: **Type 0** through **Type 3**. You can apply any of the SCD types to any column in a source table by a simple drag-and-drop operation.

- **Type 0**: is not used frequently. Some dimension data may be overwritten and other may stay unchanged over time. This is most appropriate when no effort has been made to deal with the changing dimension issues.

- **Type 1**: no history is kept in the database. New data overwrites old data. Use this type if tracking changes is not necessary. this is most appropriate when correcting certain typos, for example the spelling of a name.

- **Type 2**: the whole history is stored in the database. This type tracks historical data by inserting a new record in the dimensional table with a separate key each time a change is made. This is most appropriate to track updates, for example.

- **Type 3**: only the information about a previous value of a dimension is written into the database. This type tracks changes using separate columns. This is most appropriate to track only the previous value of a changing column.

SCD **Type 2** principle lies in the fact that a new record is added to the SCD table when changes are detected on the columns defined. Note that although several changes may be made to the same record on various columns defined as SCD **Type 2**, only one additional line tracks these changes in the SCD table.

The SCD schema in this type should include SCD-specific extra columns that hold standard log information such as:

- **start**: adds a column to your SCD schema to hold the start date. You can select one of the input schema columns as a start date in the SCD table.

- **end**: adds a column to your SCD schema to hold the end date value for a record. When the record is currently active, the end date is **NULL** or you can select **Fixed Year Value** and fill in a fictive year to avoid having a null value in the end date field.

- **version**: adds a column to your SCD schema to hold the version number of the record.

- **active**: adds a column to your SCD schema to hold the **true** or **false** status value. This column helps to easily spot the active record.
Tracking data changes using Slowly Changing Dimensions (type 0 through type 3)

This scenario describes a Job that stores and manages both the current and historical employee data in a MySQL table using SCD (Slowly Changing Dimensions).

The input data contains various employee details including name, age, role, and salary, and another id column is added to help ensuring the unicity of the input data.

At first, the following employee data is inserted to a new MySQL table using SCD:

```
1;Mark Smith;30;tester;11000.00
2;Thomas Johnson;32;developer;12000.00
3;Teddy Brown;33;tester;13000.00
```

Then the table is updated using SCD with the following renewed employee data.

```
1;Mark Smith;31;tester;11000.00
2;Thomas Johnson;32;developer;12000.00
3;Teddy Brown;33;writer;13500.00
```

You can see the age of Mark Smith is updated from 30 to 31, the role of Teddy Brown is changed from tester to writer, and his salary is raised from 13000.00 to 13500.00. In this scenario,

- we don’t want to track the data changes for the name field, so we will perform Type 0 SCD on it,
- we want the new age data to overwrite the existing data, so we will perform Type 1 SCD on it,
- we want to retain the full history of the role data, and always create a new record with the changed data and close the previous record, so we will perform Type 2 SCD on it,
- we want to keep the current and previous dimension values for the salary field, so we will perform Type 3 SCD on it.

For more information about SCD types, see SCD management methodology on page 2566.

Creating a Job for tracking data changes in MySQL using SCD

Create a Job to open a connection to a MySQL database, insert the employee data into a MySQL database table using SCD (Slowly Changing Dimensions)

This Job retrieves and displays the inserted data on the console, then updates the employee data in MySQL using SCD, retrieves and displays the updated data on the console, finally closes the connection to the MySQL database.
Procedure

1. Create a new Job and add a **tMysqlConnection** component, two **tFixedFlowInput** components, two **tMysqlSCD** components, two **tMysqlInput** components, two **tLogRow** components, and a **tMysqlClose** component by typing their names in the design workspace or dropping them from the Palette.

2. Link the first **tFixedFlowInput** component to the first **tMysqlSCD** component using a **Row > Main** connection.

3. Do the same to link the first **tMysqlInput** component to the first **tLogRow** component, the second **tFixedFlowInput** component to the second **tMysqlSCD** component, and the second **tMysqlInput** component to the second **tLogRow** component.

4. Link the **tMysqlConnection** component to the first **tFixedFlowInput** component using a **Trigger > OnSubjobOk** connection.

5. Do the same to link the first **tFixedFlowInput** component to the first **tMysqlInput** component, the first **tMysqlInput** component to the second **tFixedFlowInput** component, the second **tFixedFlowInput** component to the second **tMysqlInput** component, and the second **tMysqlInput** component to the **tMysqlClose** component.
Opening a connection to a MySQL database

Configure the `tMysqlConnection` component to open a connection to a MySQL database.

**Procedure**

1. Double-click the `tMysqlConnection` component to open its **Basic settings** view.

```
<table>
<thead>
<tr>
<th>Property Type</th>
<th>DB Version</th>
<th>Host</th>
<th>Port</th>
<th>Database</th>
<th>Username</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-In</td>
<td>Mysql 5</td>
<td>&quot;192.168.32.68&quot;</td>
<td>&quot;3306&quot;</td>
<td>&quot;talend&quot;</td>
<td>&quot;root&quot;</td>
<td>**********</td>
</tr>
</tbody>
</table>
```

2. In the **Host**, **Port**, **Database**, **Username**, and **Password** fields, enter the information required for the connection to the MySQL database.

Inserting the employee data into a MySQL table using SCD

Configure the first `tFixedFlowInput` component and the first `tMysqlSCD` component to insert the employee data into a MySQL database table using SCD (Slowly Changing Dimensions).

**Procedure**

1. Double-click the first `tFixedFlowInput` component to open its **Basic settings** view.

```
[Content]
1: Mark Smith, 30; tester; 11000.00
2: Thomas Johnson, 32; developer; 12000.00
3: Teddy Brown, 33; tester; 13000.00
```

2. Click the button next to **Edit schema** and in the pop-up window define the schema by adding five columns: `id` and `age` of Integer type, `name` and `role` of String type, and `salary` of Double type.
When done, click OK to save the changes. In the pop-up dialog box, click Yes to propagate the schema to the next component.

3. In the Mode area, select Use Inline Content (delimited file). Then in the Content field displayed, enter the following input employee data.

1;Mark Smith;30;tester;11000.00
2;Thomas Johnson;32;developer;12000.00
3;Teddy Brown;33;tester;13000.00

4. Click the first tMysqlSCD component to open its Basic settings view.

5. Select the Use an existing connection check box and from the Component List drop-down list displayed, select the connection component you have configured.

6. In the Table field, enter the name of employee_scd.

7. Click the button next to SCD Editor to open the SCD editor. All schema columns are listed on the Unused panel.

8. In the name field on the Surrogate keys panel, enter the name for the surrogate key, SK in this example.

9. From the Unused panel, drag and drop
   - id to the Source keys panel to use it as the key to ensure the unicity of the input data,
• **name** to the **Type 0 fields** panel and no special action will be performed upon dimension changes of it,

• **age** to the **Type 1 fields** panel to perform Type 1 SCD on it,

• **role** to the **Type 2 fields** panel to perform Type 2 SCD on it, and

• **salary** to the **Type 3 fields** panel to perform Type 3 SCD on it.

10. In the **Versioning** panel, select the **version** check box to hold the version numbers for the historical and current records in the SCD table, and select also the **active** check box to add the column that will hold the **True** value for the current active record or the **False** value for the historical records in the SCD table.

   When done, click **OK** to save the changes and close the SCD editor.

**Retrieving the inserted employee data from MySQL**

Configure the first **tMysqlInput** component and the first **tLogRow** component to retrieve the inserted employee data from the MySQL SCD table and display it on the console.

**Procedure**

1. Double-click the **tMysqlInput** component to open its **Basic settings** view.
2. Select the **Use an existing connection** check box and from the **Component List** drop-down list displayed, select the connection component you have configured.

3. In the **Table Name** field, enter the name of the table from which the employee data will be retrieved. In this example, it is `employee_scd`.

4. Open the schema editor of the **tMysqlSCD** component, select and copy all its schema columns, then open the schema editor of the **tMysqlInput** component, define its schema by pasting those schema columns you have copied from the **tMysqlSCD** component.

When done, click **OK** to save the changes. In the pop-up dialog box, click **Yes** to propagate the schema to the next component.

5. Click the **Guess Query** button to fill the **Query** field with the automatically generated SQL query that will extract data of all columns from the specified table.
6. Double-click the first `tLogRow` component to open its Basic settings view. In the Mode area, select Table (print values in cells of a table) for better readability of the result displayed on the console.

**Updating the employee data in MySQL using SCD**

Configure the second `tFixedFlowInput` component and the second `tMysqlSCD` component to update the employee data in MySQL using SCD (Slowly Changing Dimensions).

**Procedure**

1. Double-click the second `tFixedFlowInput` component to open its Basic settings view.
2. Click the button next to Edit schema and in the pop-up window define the schema by copying and pasting the schema of the first `tFixedFlowInput` component.
3. In the Mode area, select Use Inline Content (delimited file). Then in the Content field displayed, enter the following updated employee data.

   1;Mark Smith;31;tester;11000.00
   2;Thomas Johnson;32;developer;12000.00
   3;Teddy Brown;33;writer;13500.00

4. Repeat 4 on page 2572 through 10 on page 2573 in the procedure Inserting the employee data into a MySQL table using SCD on page 2571 to configure the second `tMysqlSCD` component.

**Retrieving the updated employee data from MySQL**

Configure the second `tMysqlInput` component and the second `tLogRow` component to retrieve the updated employee data from the MySQL SCD table and display it on the console.

**Procedure**

1. Repeat 1 on page 2573 through 5 on page 2574 in the procedure Retrieving the inserted employee data from MySQL on page 2573 to configure the second `tMysqlInput` component.
2. Repeat 6 on page 2575 in the procedure Retrieving the inserted employee data from MySQL on page 2573 to configure the second `tLogRow` component.

**Closing the connection to the MySQL database**

Configure the `tMysqlClose` component to close the connection to the MySQL database.

**Procedure**

1. Double-click the `tMysqlClose` component to open its Basic settings view.
2. From the Component List drop-down list, select the connection component that opens the connection you want to close. In this example, it is `tMysqlConnection_1`.

**Executing the Job to track data changes in MySQL using SCD**

After setting up the Job and configuring the components used in the Job for tracking data changes in MySQL using SCD, you can then execute the Job and verify the Job execution result.

**Procedure**

1. Press Ctrl + S to save the Job.
2. Press F6 to execute the Job.

As shown above, the new age data 31 for Mark Smith the old age data 30, a new record with the SK value set to 4 is created for the role change for Teddy Brown from tester to writer, and the values of the previous salary and the current salary for Teddy Brown are both kept in the newly created record.
tMysqlSCDELT

Reflects and tracks changes in a dedicated MySQL SCD table through SQL queries.

tMysqlSCDELT addresses Slowly Changing Dimension needs through SQL queries (server-side processing mode), and logs the changes into a dedicated MySQL SCD table.

tMysqlSCDELT Standard properties

These properties are used to configure tMysqlSCDELT running in the Standard Job framework.
The Standard tMysqlSCDELT component belongs to the Business Intelligence and the Databases families.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally. Enter properties manually.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file where Properties are stored. The fields that come after are pre-filled in using the fetched data.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the Mysql version you are using.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

Note:
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>The IP address of the database server.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>User authentication data for a dedicated database. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Source table</strong></td>
<td>Name of the input MySQL SCD table.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
</tbody>
</table>
| **Action on table** | Select to perform one of the following operations on the table defined:  
**None**: No action carried out on the table.  
**Drop and create the table**: The table is removed and created again  
**Create a table**: A new table gets created.  
**Create a table if not exists**: A table gets created if it does not exist.  
**Clear a table**: The table content is deleted. You have the possibility to rollback the operation.  
**Truncate a table**: The table content is deleted. You don't have the possibility to rollback the operation. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No**. |
upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see [Talend Studio User Guide](#).

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see [Talend Studio User Guide](#).

### Surrogate Key
Select the surrogate key column from the list.

### Creation
Select the method to be used for the surrogate key generation.

For more information regarding the creation methods, see [SCD management methodology](#) on page 2566.

### Source Keys
Select one or more columns to be used as keys, to ensure the unicity of incoming data.

### Use SCD Type 1 fields
Use type 1 if tracking changes is not necessary. SCD Type 1 should be used for typos corrections for example. Select the columns of the schema that will be checked for changes.

### Use SCD Type 2 fields
Use type 2 if changes need to be tracked down. SCD Type 2 should be used to trace updates for example. Select the columns of the schema that will be checked for changes.

- **Start date**: Adds a column to your SCD schema to hold the start date value. You can select one of the input schema columns as Start Date in the SCD table.
- **End Date**: Adds a column to your SCD schema to hold the end date value for the record. When the record is currently active, the End Date column shows a null value, or you can select [Fixed Year value](#) and fill it in with a fictive year to avoid having a null value in the End Date field.
- **Log Active Status**: Adds a column to your SCD schema to hold the true or false status value. This column helps to easily spot the active record.
- **Log versions**: Adds a column to your SCD schema to hold the version number of the record.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Debug mode</strong></th>
<th>Select this check box to display each step during processing entries in a database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
## Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide.* |

## Usage

| Usage rule | This component is used as an output component. It requires an input component and Row main link as input. |

| Dynamic settings | Click the [+ ] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide.* |

## Related Scenario

For related topics, see: *tMysqlSCD* on page 2563 and *Tracking data changes using Slowly Changing Dimensions (type 0 through type 3)* on page 2569.
tMysqlSP

Calls a MySQL database stored procedure.

tMysqlSP offers a convenient way to centralize multiple or complex queries in a database and call them easily.

**tMysqlSP Standard properties**

These properties are used to configure tMysqlSP running in the Standard Job framework.

The Standard tMysqlSP component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
</tbody>
</table>
| **Username** and **Password** | DB user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Schema** and **Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. |
| **Built-In**              | You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide. |
Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>SP Name</th>
<th>Type in the exact name of the Stored Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is Function / Return result in</strong></td>
<td>Select this check box, if a value only is to be returned. Select on the list the schema column, the value to be returned is based on.</td>
</tr>
</tbody>
</table>

**Parameters**

Click the Plus button and select the various **Schema Columns** that will be required by the procedures. Note that the SP schema can hold more columns than there are parameters used in the procedure.

Select the **Type** of parameter:

- **IN**: Input parameter.
- **OUT**: Output parameter/return value.
- **IN OUT**: Input parameters is to be returned as value, likely after modification through the procedure (function).
- **RECORDSET**: Input parameters is to be returned as a set of values, rather than single value.

**Note:**

Check [Scenario: Inserting data in mother/daughter tables](#) on page 2476 if you want to analyze a set of records from a database table or DB query and return single records.

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as intermediary component. It can be used as start component but only input parameters are thus allowed.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamic settings</strong></td>
<td>Click the [+] button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose</td>
</tr>
</tbody>
</table>
your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

| Limitation | The Stored Procedures syntax should match the Database syntax. |

**Scenario: Using tMysqlSP to find a State Label using a stored procedure**

The following job aims at finding the State labels matching the odd State IDs in a Mysql two-column table. A stored procedure is used to carry out this operation.

- Drag and drop the following components used in this example: tRowGenerator, tMysqlSP, tLogRow.
- Connect the components using the Row Main link.
- The tRowGenerator is used to generate the odd id number. Double-click on the component to launch the editor.
Click on the **Plus** button to add a column to the schema to generate.

Select the **Key** check box and define the **Type** to **Int**.

The **Length** equals to 2 digits max.

Use the preset function called **sequence** but customize the Parameters in the lower part of the window.

- Change the **Value** of **step** from 1 to 2 for this example, still starting from 1.
- Set the **Number of generated rows** to 25 in order for all the odd State id (of 50 states) to be generated.
- **Click OK** to validate the configuration.
- Then select the **tMysqlSP** component and define its properties.
Set the **Property type** field to **Repository** and select the relevant entry on the list. The connection details get filled in automatically.

Else, set manually the connection information.

Click **Sync Column** to retrieve the generated schema from the preceding component.

Then click **Edit Schema** and add an extra column to hold the State Label to be output, in addition to the ID.

Type in the name of the procedure in the **SP Name** field as it is called in the Database. In this example, **getstate**. The procedure to be executed states as follows:

```
DROP PROCEDURE IF EXISTS `talend`.`getstate` $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `getstate`(IN pid INT, OUT pstate VARCHAR(50))
BEGIN
    SELECT LabelState INTO pstate FROM us_states WHERE idState = pid;
END $$
```

In the **Parameters** area, click the plus button to add a line to the table.

Set the **Column** field to **ID**, and the **Type** field to **IN** as it will be given as input parameter to the procedure.

Add a second line and set the **Column** field to **State** and the **Type** to **Out** as this is the output parameter to be returned.

Eventually, set the **tLogRow** component properties.
Synchronize the schema with the preceding component.

- And select the **Print values in cells of a table** check box for reading convenience.

- Then save your Job and execute it.

The output shows the state labels corresponding to the odd state ids as defined in the procedure.

**Note:**

Check *Scenario: Inserting data in mother/daughter tables* on page 2476 if you want to analyze a set of records from a database table or DB query and return single records.

**Related scenarios**

For related scenarios, see:

- *Scenario: Retrieving personal information using a stored procedure* on page 2452.
- *Scenario: Checking number format using a stored procedure* on page 2809.
- *Executing a stored procedure using tMDMSP* on page 2190.
tMysqlTableList

Lists the names of a given set of Mysql tables using a select statement based on a Where clause. tMysqlTableList iterates on a set of table names through a defined Mysql connection.

**tMysqlTableList Standard properties**

These properties are used to configure tMysqlTableList running in the Standard Job framework.

The Standard tMysqlTableList component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <a href="#">Apply</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tMysqlConnection component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Where clause for table name selection</td>
<td>Enter the Where clause to identify the tables to iterate on.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>tMysqlTableList properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENT_TABLE: the name of the table currently iterated upon. This is a Flow variable and it returns a string.</td>
<td></td>
</tr>
<tr>
<td>NB_TABLE: the number of tables iterated upon so far. This is a Flow variable and it returns an integer.</td>
<td></td>
</tr>
<tr>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
<td></td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with Mysql components, especially with tMysqlConnection.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. 

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

---

**Scenario: Iterating on DB tables and deleting their content using a user-defined SQL template**

The following Java scenario creates a three-component job that iterates on given table names from a MySQL database using a WHERE clause. It then deletes the content of the tables directly on the DBMS using a user-defined SQL template.

For advanced use, start with creating a connection to the database that contains the tables you want to empty of their content.

- In the Repository tree view, expand Metadata and right click DB Connections to create a connection to the relevant database and to store the connection information locally.

  For more information about Metadata, see Talend Studio User Guide.

  Otherwise, drop a tMySQLConnection component in the design workspace and fill the connection details manually.

- Drop the database connection you created from the Repository onto the design workspace.

  The [Components] dialog box displays.

- Select tMysqlTableList and click OK.
The `tMysqlConnection` component displays on the design workspace with all connection details automatically filled in its **Basic settings** view.

- Drop the following two components from the Palette onto the design workspace: `tMysqlTableList` and `tELT`.
- Connect `tMysqlConnection` to `tMysqlTableList` using an **OnSubjobOk** link.

![Design Workspace Diagram](image)

- Connect `tMysqlTableList` to `tELT` using an **Iterate** link.
- If needed, double-click `tMysqlConnection` to display its **Basic settings** view and verify the connection details.

![Basic Settings](image)

In this example, we want to connect to a MySQL database called `examples`.

- In the design workspace, double-click `tMysqlTableList` to display its **Basic settings** view and define its settings.

![Basic Settings](image)

- On the **Component list**, select the relevant MySQL connection component if more than one connection is used.
- Enter a WHERE clause using the right syntax in the corresponding field to iterate on the table name(s) you want to delete the content of.

  In this scenario, we want the job to iterate on all the tables which names start with "ex".

  ![Basic Settings](image)

- In the design workspace, double-click `tELT` to display its **Basic settings** view and define its settings.
In **Database Name**, enter the name of the database containing the tables you want to process.

On the **Component list**, select the relevant MySQL connection component if more than one connection is used.

Click in the **Table name** field and press **Ctrl+Space** to access the global variable list.

From the global variable list, select `((String)globalMap.get("tMysqlTableList_1_CURRENT_TABLE"))`.

**To create the user-defined SQL template:**

- In the **Repository** tree view, expand **SQL Templates** and **MySQL** in succession.

- Right-click **UserDefined** and select **Create SQLTemplate** from the drop-down list.

The **[New SQLTemplate]** wizard opens.
• Enter a name for the new SQL template and fill in the other fields if needed and then click **Finish** to close the wizard.

An SQL pattern editor opens on the design workspace.

• Delete the existing code and enter the code necessary to carry out the desired action, deleting the content of all tables which names start with "ex" in this example.

```
1 DELETE FROM `<% = __TABLE_NAME__ %>`;
```

**Note:**

In the SQL template code, you must use the correct variable name attached to the table name parameter ("__TABLE_NAME__" in this example). To display the variable name used, put your pointer in the **Table Name** field in the basic settings of the tELT component.

```java
Database Name: "examples"
Table Name: (String)globalsMap.get("mysqlTableList_1_CURRENT_TABLE")
Schema: The variable attached to this parameter is: __TABLE_NAME__
```

• Press **Ctrl+S** to save the new user-defined SQL template.

The next step is to add the new user-defined SQL template to the SQL template list in the tELT component.

**To add the user-defined SQL template to the SQL template list:**
• In the Component view of tELT, click the SQL Templates tab to display the SQLTemplate List.

  ![SQL Template List](image)

  • Click the Add button and add two SQL template lines.
  • Click in the first line to display a drop-down arrow and then click the arrow to display the SQL template list.

  ![SQL Template List](image)

  • Select in the list the user-defined SQL template you already created.
  • Make sure that the SQL template in the second line is Commit.
  • Save your job and press F6 to execute it.

All tables in the MySQL examples database which names begin with "ex" are emptied from their content.

**Related scenario**

For tMysqlTableList related scenario, see Scenario: Iterating on a DB table and listing its column names on page 2470.
tNamedPipeClose

Closes a named-pipe at the end of a process.

tNamedPipeClose closes a named-pipe opened with tNamedPipeOpen at the end of a process.

**tNamedPipeClose Standard properties**

These properties are used to configure tNamedPipeClose running in the Standard Job framework.
The Standard tNamedPipeClose component belongs to the File family.
The component in this framework is generally available.

**Basic settings**

| Pipe | Select an existing named-pipe from the list to close. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

**Usage**

| Usage rule | This component is usually used to close a named-pipe at the end of a Job. |
| Dynamic settings | Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your pipe connection dynamically from multiple connections planned in your Job. When a dynamic parameter is defined, the Pipe box in the Basic settings view becomes unusable. |
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

**Related scenario**

For a related scenario, see Scenario: Writing and loading data through a named-pipe on page 2599.
tNamedPipeOpen

Opens a named-pipe for writing data into it.
Used in inner-process communication, tNamedPipeOpen opens a named-pipe for writing data into it.

**tNamedPipeOpen Standard properties**

These properties are used to configure tNamedPipeOpen running in the Standard Job framework.
The Standard tNamedPipeOpen component belongs to the File family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Name</th>
<th>Fill in the field with the name of the named-pipe.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Delete if already exist</strong></td>
<td>Select this check box to avoid duplicate named-pipe.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</th>
</tr>
</thead>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>PIPE_NAME: the name of the named-pipe. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PIPE_NATIVE_NAME: the native name of the named-pipe. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>PIPE_OUTPUTSTREAM: the output stream of the named-pipe. This is a Flow variable and it returns an object.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

2595
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is usually used as the starting component in an inner-process communication Job.</th>
</tr>
</thead>
</table>

Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Related scenario

For a related scenario, see Scenario: Writing and loading data through a named-pipe on page 2599.
tNamedPipeOutput

Writes data into an existing open named-pipe.
tNamedPipeOutput writes data into a named-pipe opened with tNamedPipeOpen.

**tNamedPipeOutput Standard properties**

These properties are used to configure tNamedPipeOutput running in the Standard Job framework.
The Standard tNamedPipeOutput component belongs to the File family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use existing pipe connection</th>
<th>Select this check box to use an existing named-pipe in the Pipe component list, or clear this check box to specify a named-pipe in Pipe name field.</th>
</tr>
</thead>
</table>
| Pipe component                | Select an existing named-pipe component from the list.  
                                 **Note:**  
                                 This check box will display only when you select Use existing pipe connection. |
| Pipe name                     | Fill in the field with the name of an existing named-pipe.  
                                 **Note:**  
                                 This check box will display only when you clear Use existing pipe connection. |
| Row separator                 | String (ex: "\n" on Unix) to distinguish rows in the output file.  
                                 **Note:**  
                                 This check box will display only when you clear Use existing pipe connection. |
| Field separator               | Character, string or regular expression to separate fields of the output file.  
| CSV options                   | Select this check box to take into account all parameters specific to CSV files, in particular Escape char and Text enclosure parameters. |
| Schema and Edit Schema        | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository.  
                                 Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
                                 • View schema: choose this option to view the schema only.  
                                 • Change to built-in property: choose this option to change the schema to Built-in for local changes. |
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

**Built-in**: The schema will be created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job flowcharts. Related topic: see *Talend Studio User Guide*.

**Delete pipe if it exists**

Select this check box to avoid duplicate named-pipe.

## Advanced settings

<table>
<thead>
<tr>
<th>Boolean type</th>
<th>Select a boolean type from the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

## Global Variables

**Global Variables**

- **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is a Flow variable and it returns an integer.
- **PIPE_NAME**: the name of the named-pipe. This is a Flow variable and it returns a string.
- **PIPE_NATIVE_NAME**: the native name of the named-pipe. This is a Flow variable and it returns a string.
- **PIPE_OUTPUTSTREAM**: the output stream of the named-pipe. This is a Flow variable and it returns an object.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*. 
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is usually connected to another component in a subjob that reads data from a source.</th>
</tr>
</thead>
</table>

Dynamic settings

Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your pipe connection dynamically from multiple connections planned in your Job.

The Dynamic settings table is available only when the Use existing pipe connection check box is selected in the Basic settings view. When a dynamic parameter is defined, the Pipe component list box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Scenario: Writing and loading data through a named-pipe

The following scenario creates a multi-component Job, which writes data into an open named-pipe and displays the data onto the console.
Dropping and linking the components

Procedure
1. Drop the following components from the Palette to the design workspace: **tNamedPipeOpen**, **tParallelize**, **tNamedPipeClose**, **tFileInputDelimited**, **tSleep**, **tLogRow**, **tRowGenerator** and **tNamedPipeOutput**.
2. Connect **tNamedPipeOpen** to **tParallelize** using a **Trigger > OnSubjobOk** connection.
3. Connect **tParallelize** to **tFileInputDelimited** using a **Trigger > Parallelize** connection.
4. Connect **tParallelize** to **tSleep** using a **Trigger > Parallelize** connection.
5. Connect **tFileInputDelimited** to **tLogRow** using a **Row > Main** connection.
6. Connect **tParallelize** to **tNamedPipeClose** using a **Trigger > Synchronize (Wait for all)** connection.
7. Connect **tSleep** to **tRowGenerator** using a **Trigger > OnComponentOk** connection.
8. Connect **tRowGenerator** to **tNamedPipeOutput** using a **Row > Main** connection.

Configuring the components

Configuring the input component

Procedure
1. Double-click **tNamedPipeOpen** to define its properties in its **Basic settings** view.
   Fill in the **Name** field with the name of a named-pipe and select **Delete if already exist** to avoid duplicate named-pipe.
2. Double-click **tParallelize** to define its properties in its **Basic settings** view.
Select **end of all subjobs** from the **Wait for** list.
Fill in the **Sleep Duration** field with **100** to set the sleep duration.

3. Double-click **tFileInputDelimited** to define its properties in its **Basic settings** view.
Fill in the **File name/Stream** field with the following expression to use the name of the existing named-pipe defined in the **Basic settings** view of **tNamedPipeOpen**:

```
((String)globalMap.get("tNamedPipeOpen_1_PIPE_NATIVE_NAME"))
```

4. Click the three-dot button next to **Edit schema**.
5. Click the plus button to add three columns for `tFileInputDelimited`. Fill the three Column fields with `id`, `first_name` and `last_name` and set the Type of `id` to `Integer`. Keep the rest of the settings as default.

6. Click **OK** to save the settings for the schema.

7. Keep the rest of the settings in the **Basic settings** view of `tFileInputDelimited` as default.

8. Double-click `tSleep` and fill the **Pause (in seconds)** field with 1.

9. Double-click `tRowGenerator` to define its properties in its **Basic settings** view.

10. Click **RowGenerator Editor** to define the schema.

11. Click the plus button to add three columns for `tRowGenerator`. Fill the three Column fields with `id`, `first_name` and `last_name` and set the Type of `id` to `Integer`. Keep the rest of the settings of Type as default.

12. Select `sequence` from the list in the Functions field for `id`.

13. Select `getFirstName` from the list in the Functions field for Column `first_name`. 
14. Select `TalendDataGenerator.getLastName` from the list in the **Functions** field for Column `last_name`.

15. Select `id`, fill the **Value** field under **Function parameters** tab with `s1` for **sequence identifier**, `1001` for **start value** and `1` for **step**.

16. Click **OK** to save the settings.

**Configuring the output component**

**Procedure**

1. Double-click **tNamedPipeOutput** to define its properties in its **Basic settings** view.

2. Select the **Use existing pipe connection** check box and select **tNamedPipeOpen_1** from the **Pipe component** list.

3. Select **Delete pipe if it exists** to avoid duplicate named-pipe.

4. Click **Sync columns** to retrieve the schema from the preceding component.

5. Leave the rest of the settings as they are.

6. Double-click **tLogRow** to define its properties in its **Basic settings** view.

7. Click **Sync columns** to retrieve the schema from the preceding component.

8. Select **Table** in the **Mode** area.
9. Double-click **tNamedPipeClose** to define its properties in its **Basic settings** view.

10. Select **tNamedPipeOpen_1** from the **Pipe** list.

## Saving and executing the Job

### Procedure

Press **F6** to execute the Job.

```
Starting job namedPipe_Sample_01 at 18:13 19/05/2011.
[statistics] connecting to socket on port 3042
[statistics] connected
[----------------------]
<table>
<thead>
<tr>
<th>id</th>
<th>first_name</th>
<th>last_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Bill</td>
<td>Cleveland</td>
</tr>
</tbody>
</table>
[----------------------]
[statistics] disconnected
Job namedPipe_Sample_01 ended at 18:13 19/05/2011. [exit code=0]
```

### Results

The data written into the named-pipe is displayed onto the console.
tNeo4jBatchOutput

Receives data from the preceding component and writes the data into a local Neo4j database.
tNeo4jBatchOutput is used to write nodes in a Neo4j database based on the defined index.

**tNeo4jBatchOutput Standard properties**

These properties are used to configure tNeo4jBatchOutput running in the Standard Job framework.
The Standard tNeo4jBatchOutput component belongs to the Big Data and the Databases families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. This component supports Neo4j version V3.2.X only and does not support the remote mode. Therefore, do not reuse the connection to versions other than V3.2.X defined in a tNeo4jConnection component and do not select the Remote server check box in tNeo4jConnection. Do not use a 2.X.X version and a 3.X.X version in the same Job. Otherwise, class conflict issues occur.</td>
</tr>
<tr>
<td><strong>Database path</strong></td>
<td>Specify the directory to hold your data files. If the specified directory does not exist, it will be created. This field appears only if you do not select the Use an existing connection check box.</td>
</tr>
<tr>
<td><strong>Shutdown after job</strong></td>
<td>Select this check box to shutdown the Neo4j database connection when no more operations on Neo4j are going to be performed after the current component. Alternatively, you can use tNeo4jClose to shutdown the database. This avoids errors such as “Id file not properly shutdown” at next execution of Jobs involving Neo4j. This check box is available only if the Use an existing connection check box is not selected.</td>
</tr>
<tr>
<td><strong>Field that contains the label list</strong></td>
<td>Select the column from the input schema you have defined in the preceding components to provide labels for the nodes to be created. When the input data provides multiple labels for a node, separate those labels by semi-colons (;) in the preceding components.</td>
</tr>
<tr>
<td><strong>Index name</strong></td>
<td>Enter, within double quotation marks, the name of the index to be created for the nodes. This index is used</td>
</tr>
</tbody>
</table>
to store the identifiers of the nodes and help create relationships.

**Import identifier**
Select the column from the input schema you have defined in the preceding components to provide identifiers for the nodes to be created. These identifiers are stored in the index to be created and must be unique across the index.

Therefore, ensure that the column to be used contains only unique values.

**Save the identifier**
Select this check box to save the technical identifiers (native graph ids).

For further information about the different types of Neo4j entity identifiers, see Neo4j documentation: Entity identifier.

**Number of index elements in memory**
Enter, without double quotation marks, the cache size used to keep the index elements in memory. This value must be enough to keep all of the index elements in memory.

For further information, see Neo4j documentation: Memory tuning.

---

### Advanced settings

**Neo4j configuration**
Add parameters to the table to configure the database to be created.

For further information, see Neo4j documentation: Configuration settings.

When entering values, use the syntax demonstrated by the examples given alongside the column names of this Nodes files table.

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

---

### Global Variables

**Global Variables**

**NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | This component is used as an output component and it always needs an incoming link. |
tNeo4jBatchOutputRelationship

Receives data from the preceding component and writes relationships in bulk into a local Neo4j database.

tNeo4jBatchOutputRelationship Standard properties

These properties are used to configure tNeo4jBatchOutputRelationship running in the Standard Job framework.

The Standard tNeo4jBatchOutputRelationship component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

| Use existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. This component supports Neo4j version V3.2.X only and does not support the remote mode. Therefore, do not reuse the connection to versions other than V3.2.X defined in a tNeo4jConnection component and do not select the Remote server check box in tNeo4jConnection. Do not use a 2.X.X version and a 3.X.X version in the same Job. Otherwise, class conflict issues occur. |
| Database path | Specify the directory to hold your data files. This field appears only if you do not select the Use an existing connection check box. |
| Shutdown after job | Select this check box to shutdown the Neo4j database connection when no more operations on Neo4j are going to be performed after the current component. Alternatively, you can use tNeo4jClose to shutdown the database. This avoids errors such as “Id file not properly shutdown” at next execution of Jobs involving Neo4j. This check box is available only if the Use an existing connection check box is not selected. |
| Field for relationship types | Select the column from the input schema you have defined in the preceding components to provide types for the relationships to be created. |
| Direction of the relationship | Select the direction of the relationships to be created: • Outgoing: The relationship starts from the start node to the end node. |
### Start node of the relationship
Defining the start node of each relationship using the node identifier:
- **Name of the batch index:** select the tNeo4jBatchOutput component used to create the start nodes. The name of the index for these nodes are retrieved from that component.
- **Field name for the batch index:** select the column from the input schema you have defined in the preceding components to provide the name of the start node of each relationship to be created.

### End node of the relationship
Defining the end node of each relationship using the node identifier:
- **Name of the batch index:** select the tNeo4jBatchOutput component used to create the end nodes. The name of the index for these nodes are retrieved from that component.
- **Field name for the batch index:** select the column from the input schema you have defined in the preceding components to provide the name of the end node of each relationship to be created.

### Die on error
Select the check box to stop the execution of the Job when an error occurs.
Clear the check box to skip any rows on error and complete the process for error-free rows.

### Advanced settings

#### Neo4j configuration
Add parameters to the table to configure the database to be created.
For further information, see Neo4j documentation: Configuration settings.
When entering values, use the syntax demonstrated by the examples given alongside the column names of this Nodes files table.

#### tStatCatcher Statistics
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

#### Global Variables
**NB_LINE:** the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

**ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions...
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

## Usage

| Usage rule | This component is used as an output component and it always needs an incoming link. |

## Writing information of actors and movies to Neo4j with hierarchical relationship using Neo4j Batch components

In this scenario, Neo4j Batch components are used to import data about actors and movies from two CSV files in a local Neo4j database and create relationship for the data based on another CSV file that describes the actors' roles in the movies.

This scenario applies only to a Talend solution with Big Data.

The Neo4j Batch components provided by Talend supports bulk writing to a local Neo4j database only. They can be used neither with Neo4j versions prior to V3.2.X nor alongside Neo4j components that are using one of those Neo4j versions.

The components to be used are:

- One tNeo4jConnection component: it opens the connection to Neo4j to be reused.
- Three tFileInputDelimited components: they read the input information of actors and movies.
- Two tNeo4jBatchOutput components: they write information of movies and actors to the connected Neo4j database.
- One tNeo4jBatchOutputRelationship component: it creates relationship between actors and movies.
- One tNeo4jBatchSchema component: it creates an uniqueness constraint on the nodes in the database.
Creating the Neo4j Batch Job

Procedure

1. Ensure that the status of your Neo4j service and Neo4j console is stop.
   
   If you are using command-line to manage Neo4j, you can use `neo4j status` to check the status; if you have installed the Neo4j desktop application, you can check it directly in this application.

2. From the Repository on the Integration perspective, create a Job and add the components to be used by typing their names in the design workspace or dropping them from the Palette.

3. Connect the first `tFileInputDelimited` component to the first `tNeo4jBatchOutput` component using a Row > Main link. This Subjob imports the actors data in the Neo4j database.

4. Connect the second `tFileInputDelimited` component to the second `tNeo4jBatchOutput` component using a Row > Main link. This Subjob imports the movies data in the Neo4j database.

5. Connect the third `tFileInputDelimited` component to the `tNeo4jBatchOutputRelationship` component using a Row > Main link. This Subjob creates relationship between actors and movies.


Configuring the Neo4j connection to be reused

Procedure

1. Double-click the `tNeo4jConnection` component to open its Basic settings view.
2. From the DB Version list, select Neo4J 3.2.X.

3. Ensure that the Use a remote server check box is clear because the Neo4j Batch components work only on the local mode.

4. In the Database path field, enter the path or browse to the database file.

**Bulk-writing the actors data in Neo4j**

**Procedure**

1. Double-click the first tFileInputDelimited component to open its Component view.

2. In the File name/Stream field, enter the path or browse to the CSV file that describes the actors' IDs, names and their labels to be used in Neo4j.

   The input CSV file used in this example reads as follows:

   ```
   keanu,"Keanu Reeves",Actor
   laurence,"Laurence Fishburne",Actor
   carrieanne,"Carrie-Anne Moss",Actor
   ```

   The double quotation marks on the actor names are not mandatory.

3. Click the [...] button next to Edit schema to open the schema editor, and define the input schema based on the structure of the input file.

   In this example, the columns are id, name and label, all of type String.
4. Click OK to close this editor and accept the propagation of the schema to the next component.
5. In the Field separator field, enter a comma (,) to replace the default semicolon (;).
6. Double-click the first tNeo4jBatchOutput component to open its Component view.

7. Select the Use an existing connection check box to reuse the Neo4j database connection opened by the tNeo4jConnection component.
8. Verify that the Shutdown after Job check box is clear.
9. From the Field that contains the label list drop-down list, select the column that provides labels.
10. In the Index name field, enter the name of the index to be created for the nodes.
11. From Import identifier drop-down list, select the column that provides IDs.

**Bulk-writing the movies data into Neo4j**

**Procedure**

1. Double-click the second tFileInputDelimited component to open its Component view.
2. In the **File name/Stream** field, enter the path or browse to the CSV file that describes the movies’ IDs, names, release years and their labels to be used in Neo4j.

The input CSV file used in this example reads as follows:

```
tt0133093,"The Matrix",1999,Movie
tt0234215,"The Matrix Reloaded",2003,Movie;Sequel
tt0242653,"The Matrix Revolutions",2003,Movie;Sequel
```

The double quotation marks on the movie names are not mandate.

3. Click the [...] button next to **Edit schema** to open the schema editor, and define the input schema based on the structure of the input file.

In this example, the columns are id, title, released and label.

4. Click **OK** to close this editor and accept the propagation of the schema to the next component.

5. In the **Field separator** field, enter a comma (,) to replace the default semicolon (;).

6. Double-click the second **tNeo4jBatchOutput** component to open its **Component** view.
7. Select the **Use an existing connection** check box to reuse the Neo4j database connection opened by the tNeo4jConnection component.

8. Verify that the **Shutdown after Job** check box is clear.

9. From the **Field that contains the label list** drop-down list, select the column that provides labels.

10. In the **Index name** field, enter the name of the index to be created for the nodes.

11. From **Import identifier** drop-down list, select the column that provides IDs.

### Creating relationships in bulk

**Procedure**

1. Double-click the third **tFileInputDelimited** component to open its **Component** view.

2. In the **File name/Stream** field, enter the path or browse to the CSV file that describes the actor-movie relationships.
The input CSV file used in this example reads as follows:

```
keanu,"Neo",tt0133093,ACTED_IN
keanu,"Neo",tt0234215,ACTED_IN
keanu,"Neo",tt0242653,ACTED_IN
laurence,"Morpheus",tt0133093,ACTED_IN
laurence,"Morpheus",tt0234215,ACTED_IN
laurence,"Morpheus",tt0242653,ACTED_IN
carrieanne,"Trinity",tt0133093,ACTED_IN
carrieanne,"Trinity",tt0234215,ACTED_IN
carrieanne,"Trinity",tt0242653,ACTED_IN
```

The double quotation marks on the role names are not mandatory. The value `ACTED_IN` is an user-defined relationship type that explains the relationship between the actors and the movies.

3. Click the [...] button next to **Edit schema** to open the schema editor, and define the input schema based on the structure of the input file.

In this example, the columns are **from**, **role**, **to** and **type**.

4. Click **OK** to close this editor and accept the propagation of the schema to the next component.

5. In the **Field separator** field, enter a comma (,) to replace the default semicolon (;).

6. Double-click the **tNeo4jBatchOutputRelationship** component to open its **Component** view.
7. Select the **Use an existing connection** check box to reuse the Neo4j database connection opened by the tNeo4jConnection component.

8. Verify that the **Shutdown after Job** check box is clear.

9. From the **Field for relationship type** drop-down list, select the column that provides the relationship types.

10. From the **Direction of the relationship** drop-down list, select **Outgoing**.

11. In the **Start node of the relationship** area, select the **tNeo4jBatchOutput** component that provides the index of the start nodes, which is the `asActors` index in this example from the first tNeo4jBatchOutput. Then from the **Field name for the batch index** drop-down list, select the column that provides the actor names as the start nodes.

12. Repeat this action in the **End node of the relationship** area to select the `asMovie` index from the second tNeo4jBatchOutput and then select the column that provides the movie names as the end nodes.

**Adding uniqueness constraints on the nodes**

**Procedure**

1. Double-click the **tNeo4jBatchSchema** component to open its **Component** view.
2. Select the **Use an existing connection** check box to reuse the Neo4j database connection opened by the tNeo4jConnection component.

3. Select the **Shutdown after Job** check box to properly close the connection after the execution.

4. In the **Schema definition** table, add two rows by clicking the [+] button twice:
   a) In the **Schema type** column, select **Node property is unique** for both of the rows to add uniqueness constraints to nodes in Neo4j.

   b) In the **For node with Label** column, enter, within double quotation marks, **Actor** and **Movie** respectively, which are the labels used by the actor nodes and the movie nodes. Therefore, what you enter here must be identical with the labels previously used when creating those nodes.

   c) In the **On property** column, enter, within double quotation marks, the node properties to which you need to add uniqueness constraints. For the actor nodes, enter **name** and for the movie nodes, enter **title**. The values you enter here must be identical with the column names previously defined to provide actor names and movie names for the nodes to be created by the tNeo4jBatchOutput components.

5. Press **Ctrl+S** to save the Job, and press **F6** or click **Run** on the **Run** tab to execute the Job.

**Results**

Once the Job runs successfully to the end, check the result in your Neo4j browser:
tNeo4jBatchSchema

Defines the schema of a local Neo4j database.

**tNeo4jBatchSchema Standard properties**

These properties are used to configure tNeo4jBatchSchema running in the Standard Job framework. The Standard tNeo4jBatchSchema component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. This component supports Neo4j version V3.2.X only and does not support the remote mode. Therefore, do not reuse the connection to versions other than V3.2.X defined in a tNeo4jConnection component and do not select the Remote server check box in tNeo4jConnection. Do not use a 2.X.X version and a 3.X.X version in the same Job. Otherwise, class conflict issues occur.</td>
</tr>
<tr>
<td><strong>Database path</strong></td>
<td>Specify the directory to hold your data files. If the specified directory does not exist, it will be created. This field appears only if you do not select the <strong>Use an existing connection</strong> check box.</td>
</tr>
<tr>
<td><strong>Shutdown after job</strong></td>
<td>Select this check box to shutdown the Neo4j database connection when no more operations on Neo4j are going to be performed after the current component. Alternatively, you can use tNeo4jClose to shutdown the database. This avoids errors such as “Id file not properly shutdown” at next execution of Jobs involving Neo4j. This check box is available only if the <strong>Use an existing connection</strong> check box is not selected.</td>
</tr>
</tbody>
</table>
| **Schema definition**        | Add indexes and unique property constraints to define the schema:  
  - **Schema type**: select whether it is an index (Index on node property) or a constraint (Node property is unique) to be defined.  
  - **For node with label**: enter, within double quotation marks, the label of a node.  
  - **On property**: enter, within double quotation marks, the property of a node. |
Advanced settings

| Neo4j configuration | Add parameters to the table to configure the database to be created.  
|                     | For further information, see Neo4j documentation: Configuration settings.  
|                     | When entering values, use the syntax demonstrated by the examples given alongside the column names of this Nodes files table. |

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

Global Variables

| Global Variables | NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
|                  | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
|                  | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
|                  | To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
|                  | For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component is used standalone in a Subjob to write a Neo4j graph schema. |
tNeo4jClose

Close an active connection to an Neo4j database in embedded mode.

**tNeo4jClose Standard properties**

These properties are used to configure tNeo4jClose running in the Standard Job framework.

The Standard tNeo4jClose component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Connection | Select an active Neo4j database connection to be closed. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

**Usage**

| Usage rule | This component is to be used along with other Neo4j components, especially with tNeo4jConnection. |

| Limitation | n/a |

**Related scenarios**

For scenario in which tNeo4jClose is used, see:
• Scenario: Writing information of actors and movies to Neo4j with hierarchical relationship on page 2648,

• Scenario 1: Creating nodes with a label using a Cypher query on page 2660, and

• Scenario 2: Importing data from a CSV file to Neo4j using a Cypher query on page 2664.
tNeo4jConnection

Opens a connection to a Neo4j database to be reuse by other Neo4j components.
tNeo4jConnection starts the database in embedded mode, and checks the server availability in REST mode.

**tNeo4jConnection Standard properties**

These properties are used to configure tNeo4jConnection running in the Standard Job framework.
The Standard tNeo4jConnection component belongs to the Big Data and the Databases families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB version</strong></td>
<td>Select the Neo4j version you are using.</td>
</tr>
<tr>
<td></td>
<td>Upon selecting a database version, you will be prompted to install the</td>
</tr>
<tr>
<td></td>
<td>corresponding database driver JAR files if not yet installed. You can</td>
</tr>
<tr>
<td></td>
<td>find more details about how to install external modules in Talend Help</td>
</tr>
<tr>
<td></td>
<td>Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
<tr>
<td><strong>Remote server</strong></td>
<td>Select this check box if you use a Neo4j remote server, and specify the</td>
</tr>
<tr>
<td></td>
<td>root URL in the <strong>Server URL</strong> field.</td>
</tr>
<tr>
<td><strong>Server URL</strong></td>
<td>Specify the root URL.</td>
</tr>
<tr>
<td></td>
<td>This field is available only if the Use a remote server check box is</td>
</tr>
<tr>
<td></td>
<td>selected.</td>
</tr>
<tr>
<td><strong>Read only</strong></td>
<td>Select this check box if you want to use the embedded database in read</td>
</tr>
<tr>
<td></td>
<td>only mode. It’s useful if an application is already dedicated with the</td>
</tr>
<tr>
<td></td>
<td>database. This check box is available only when you are using a 2.X.X</td>
</tr>
<tr>
<td></td>
<td>version.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning:</strong> Do not use this mode when you have any output Neo4j component</td>
</tr>
<tr>
<td></td>
<td>in your Job such as tNeo4jOutput, tNeo4jOutputRelationship or tNeo4jRow.</td>
</tr>
<tr>
<td><strong>Database path</strong></td>
<td>If you use Neo4j in embedded mode, specify the path of data file.</td>
</tr>
<tr>
<td></td>
<td>This field is available only if the Use a remote server check box is not</td>
</tr>
<tr>
<td></td>
<td>selected.</td>
</tr>
</tbody>
</table>
Advanced settings

| tStatCatcher Statistics | Select this check box to collect the log data at a component level. |

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component is generally used with other Neo4j components, particularly tNeo4jClose. |
| Limitation | n/a |

Related scenarios

For scenario in which tNeo4jConnection is used, see:

- Scenario: Writing information of actors and movies to Neo4j with hierarchical relationship on page 2648,
- Scenario 1: Creating nodes with a label using a Cypher query on page 2660, and
- Scenario 2: Importing data from a CSV file to Neo4j using a Cypher query on page 2664.
tNeo4jImportTool

Uses Neo4j Import Tool to create a Neo4j database and import large amounts of data in bulk from CSV files to this database.

**tNeo4jImportTool Standard properties**

These properties are used to configure tNeo4jImportTool running in the Standard Job framework. The Standard tNeo4jImportTool component belongs to the Big Data and the Databases families. The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neo4j database folder</td>
<td>Enter, or browse to the directory you want tNeo4jImportTool to create a Neo4j database. The name you put in the path is used to name this database.</td>
</tr>
<tr>
<td>Nodes files</td>
<td>Complete this table to provide the CSV files that contain the information about the nodes to be imported.</td>
</tr>
<tr>
<td>Path to CSV file</td>
<td>enter, within double quotation marks, the path to the CSV file to be read.</td>
</tr>
<tr>
<td>CSV headers</td>
<td>enter, within double quotation marks, the headers used in this CSV file.</td>
</tr>
<tr>
<td>Relationship files</td>
<td>Complete this table to provide the CSV files that contain the information about the relationships to be imported.</td>
</tr>
<tr>
<td>Path to CSV file</td>
<td>enter, within double quotation marks, the path to the CSV file to be read.</td>
</tr>
<tr>
<td>CSV headers</td>
<td>enter, within double quotation marks, the headers used in this CSV file.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neo4j configuration</td>
<td>Add parameters to the table to configure the database to be created.</td>
</tr>
<tr>
<td></td>
<td>For further information, see Neo4j documentation: Configuration settings.</td>
</tr>
<tr>
<td></td>
<td>When entering values, use the syntax demonstrated by the examples given alongside the column names of this Nodes files table.</td>
</tr>
</tbody>
</table>
### Import configuration
Add parameters to the table to configure Neo4j Import Tool.
For further information, see Neo4j documentation: Import Tool usage.
When entering values, use the syntax demonstrated by the examples given alongside the column names of this Nodes files table.

### tStatCatcher Statistics
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables
**Global Variables**
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.
- A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
- To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
- For further information about variables, see *Talend Studio User Guide*.

### Usage
**Usage rule**
This component is to be used standalone in a Subjob to load data in bulk from given CSV files that have been created with specific structure.
For further information about the CSV structure required by Neo4j Import Tool, see Import.
You can use tFileOutputDelimited to create the CSV files to be used. If so, bear the following configuration in mind:
- **Line separator**: "\n"
- **Field separator**: "," 
- Do not include headers.
- File encoding: UTF-8.
tNeo4jInput

Reads data from Neo4j and sends data in the output Talend flow.

tNeo4jInput reads data from Neo4j based on Cypher query allowing any further transformation or processing of the data in the rest of the Job.

**tNeo4jInput Standard properties**

These properties are used to configure tNeo4jInput running in the Standard Job framework.
The Standard tNeo4jInput component belongs to the Big Data and the Databases families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB version</td>
<td>Select the Neo4j version you are using.\nThis list is not shown if the Use an existing connection check box is selected.\nUpon selecting a database version, you will be prompted to install the corresponding database driver JAR files if not yet installed. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
<tr>
<td>Shutdown after job</td>
<td>Select this check box to shutdown the Neo4j database connection when no more operations on Neo4j are going to be performed after the current component.\nAlternatively, you can use tNeo4jClose to shutdown the database.\nThis avoids errors such as &quot;Id file not properly shutdown&quot; at next execution of Jobs involving Neo4j.\nThis check box is available only if the Use an existing connection check box is not selected.</td>
</tr>
<tr>
<td>Remote server</td>
<td>Select this check box if you use a Neo4j remote server, and specify the root URL in the Server URL field.</td>
</tr>
<tr>
<td>Database path</td>
<td>If you use Neo4j in embedded mode, specify the path of data file.\nThis field appears only if you do not select the Use an existing connection check box or the Remote server check box.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the</td>
</tr>
</tbody>
</table>
next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Query</th>
<th>Enter your Cypher query in double quotes, with return parameters matching the mapping table.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapping</td>
<td>Complete this table to specify the column or columns to be extracted and the corresponding column family or families. The <strong>Column</strong> fields of this table are automatically filled once you have defined the schema; fill the <strong>Return parameter</strong> fields with return parameters to map the node properties in the Cypher query with the schema columns. Be sure to enclose your return parameters in double quotes.</td>
</tr>
<tr>
<td>Advanced settings</td>
<td></td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
<tr>
<td>Global Variables</td>
<td></td>
</tr>
</tbody>
</table>
| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
For further information about variables, see Talend Studio User Guide.

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component always needs an output link.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>n/a</td>
</tr>
</tbody>
</table>

**Related scenarios**

For a related scenario, see Scenario 1: Writing data to a Neo4j database and reading specific data from it on page 2635.
tNeo4jOutput

Receives data from the preceding component and writes the data into Neo4j.

tNeo4jOutput is used to write data into a Neo4j database, and/or update or delete entries in the database based on the index defined.

**tNeo4jOutput Standard properties**

These properties are used to configure tNeo4jOutput running in the Standard Job framework.

The Standard tNeo4jOutput component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB version</td>
<td>Select the Neo4j version you are using. This list is not shown if the Use an existing connection check box is selected. Upon selecting a database version, you will be prompted to install the corresponding database driver JAR files if not yet installed. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
<tr>
<td>Remote server</td>
<td>Select this check box if you use a Neo4j remote server, and specify the root URL in the Server URL field.</td>
</tr>
<tr>
<td>Database path</td>
<td>If you use Neo4j in embedded mode, specify the directory to hold your data files. If the specified directory does not exist, it will be created. This field appears only if you do not select the Use an existing connection check box or the Remote server check box.</td>
</tr>
<tr>
<td>Shutdown after job</td>
<td>Select this check box to shutdown the Neo4j database connection when no more operations on Neo4j are going to be performed after the current component. Alternatively, you can use tNeo4jClose to shutdown the database. This avoids errors such as “Id file not properly shutdown” at next execution of Jobs involving Neo4j. This check box is available only if the Use an existing connection check box is not selected.</td>
</tr>
<tr>
<td>Mapping</td>
<td>Click the [...] button or double-click the component on the design workspace to open the indexes and</td>
</tr>
</tbody>
</table>
relationships mapping editor. Use it to index node or create relationships during the node insertion.

- Select the **Auto indexed** check box for a column to automatically index nodes with this property.

- **Index creation**: With a column selected, click the [+] button to create as many indexes as you want on nodes with the property corresponding to the selected column.
  - **Name**: Specify an index name in double quotes.
  - **Key**: Specify an index key in double quotes.
  - **Value (empty for current row)**: Specify an index value in double quotes. If you leave this field empty, the default value of the index added on each node will be the value of this property of the current node.
  - **Unique**: Select this check box if you want the defined index to be created only once within the graph, rather than on each node.

- **Relationship creation**: With a column selected, click the [+] button to create as many relationships as you want for nodes with the property corresponding to the selected column.
  - **Type**: Specify a relationship type in double quotes.
  - **Direction**: Select a relationship direction, between **Outgoing** and **Incoming**.
  - **Index name**: Specify an index name for the relationship, in double quotes.
  - **Index key**: Specify an index key for the relationship, in double quotes.
  - **Value (empty for current row)**: Specify an index value for the relationship, in double quotes. If you leave this field empty, the default value of the index added on the relationship will be the value of this property of the current node.

### Use label (Neo4j > 2.0)

Select this check box to create nodes with a label. Enter your label name in the **Label name** field.

This check box is not shown if Neo4j 1.X.X is selected from the **DB Version** list or Delete is selected from the **Data action** list.

Note that this option works only with Neo4j 2.0 onwards and Java 7.

### Data action

On the data of the node, you can perform:

- **Insert**: Add new node to the database.
- **Update**: Make changes to existing entries.
- **Update or insert**: Search the node with an index to update and make changes. If the node doesn’t exist, a new node will be inserted.
- **Delete**: Remove nodes fetched by the index according to the input flow.

| **Index name** | Specify the index name to query. This field is available only if the action selected in **Data action** is other than **Insert**. |
| **Index key** | Specify the index key to query. This field is available only if the action selected in **Data action** is other than **Insert**. |
| **Index value** | Select the index value to query. This field is available only if the action selected in **Data action** is other than **Insert**. |

**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center [https://help.talend.com](https://help.talend.com).
## Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Commit every**                             | Enter the number of rows to be completed before committing batches of nodes to the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.  
**Warning:**  
This option is only supported by the embedded mode of the database. You can’t make transactions in REST mode. |
| **Batch import**                             | Select this check box to activate the batch mode.  
**Warning:**  
This option is only supported by the embedded mode of the database.  
It is recommended that you perform a backup operation before executing the Job to prevent data corruption.  
**Note:**  
If you have configured index creation on multiple columns in the Mapping table, it is recommended that you select the Unique check box in the index setting for the last column to avoid creating unwanted redundant indexes that may cause batch load issues.  
If you want more explanations about memory mapping configuration of batch import, please refer to Neo4j documentation at: http://neo4j.com/docs/stable/batchinsert-examples.html. |
| **Node store mapped memory**                 | Type in the memory size in MB allocated to nodes.                                                                                                                                                           |
| **Relationship store mapped memory**         | Type in the memory size in MB allocated to relationships.                                                                                                                                                   |
| **Property store mapped memory**             | Type in the memory size in MB allocated to property.                                                                                                                                                        |
| **String store mapped memory**               | Type in the memory size in MB allocated to strings.                                                                                                                                                         |
| **Array store mapped memory**                | Type in the memory size in MB allocated to arrays.                                                                                                                                                           |
| **tStatCatcher Statistics**                  | Select this check box to gather the Job processing metadata at the Job level as well as at each component level.                                                                                             |

## Global Variables

**Global Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions</td>
</tr>
</tbody>
</table>
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as an output component and it always needs an incoming link.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Scenario 1: Writing data to a Neo4j database and reading specific data from it

This scenario applies only to a Talend solution with Big Data.

This basic scenario describes a Job composed of two subjobs: the first subjob reads employees data from a CSV file and writes it to a Neo4j database, and then triggers the second subjob, which reads the employees data based on certain query conditions from the Neo4j database and displays the data on the Run console.

Adding and linking components

Procedure
1. Create a Job and add the following components to the Job by typing theirs names in the design workspace or dropping them from the Palette:
   - a tFileInputDelimited component, to read the employees data from a CSV file,
   - a tNeo4jOutput component to write the employees data to a Neo4j database,
   - a tNeo4jInput component to read the employees data from the Neo4j database based on given conditions, and
   - a tLogRow component to display the data on the Run console.
2. Link the tFileInputDelimited component to the tNeo4jOutput component using a Row > Main connection.
3. Link the tNeo4jInput component to the tLogRow component using a Row > Main connection.
4. Link the tFileInputDelimited component to the tNeo4jInput component using a Trigger > On Subjob Ok connection.
5. Label the components to better identify their roles in the Job.
Configuring the components

Importing data to the Neo4j database

Procedure

1. Double-click the tFileInputDelimited component to open its Basic settings view on the Components tab.

2. In the File name/Stream field, specify the path to the CSV file that contains the employees data to read.

The input CSV file used in this example is as follows:

```
employeeID;employeeName;age;hireDate;salary;managerID
1;Rutherford Roosevelt;38;06-10-2008;13336.58;m5
2;Warren Adams;43;05-22-2008;11626.68;m6
3;Andrew Roosevelt;55;04-01-2007;10052.95;m4
4;Herbert Quincy;54;06-14-2007;10694.71;m6
5;Woodrow Polk;33;08-14-2007;13751.50;m4
6;Theodore Johnson;47;01-26-2008;12426.87;m6
7;Benjamin Adams;32;02-25-2008;10438.65;m4
8;Woodrow Harrison;51;10-11-2008;11188.27;m5
9;George Truman;40;04-28-2008;14254.49;m5
10;Harry Jackson;38;04-01-2008;12798.78;m6
```
3. In the **Header** field, specify the number of rows to skip as header rows. In this example, the first row of the CSV file is the header row.

4. Click the [...] button next to **Edit schema** to open the [Schema] dialog box, and define the input schema based on the structure of the input file. In this example, the input schema is composed of six columns: **employeeID** (integer), **employeeName** (String), **age** (Integer), **hireDate** (Date), **salary** (Float), and **managerID** (String).

When done, click **OK** to close the [Schema] dialog box and propagate the schema to the next component.

5. Click the tNeo4jOutput component and select the **Component** tab to open its **Basic settings** view.

6. Define a Neo4j database connection. In this example, the Neo4j database is accessible in REST mode, so select the **Remote server** check box and specify the URL of the Neo4j server in the **Server URL** field, “http://localhost:7474/db/data” in this example.

7. If needed, click the **Sync columns** button to ensure the component has the same schema as the preceding component.

Keep the rest of the parameters as they are.
Reading data from the Neo4j database

Procedure

1. Double-click the tNeo4jInput component to open its Basic settings view.

   ![Basic settings view](image)

   - Connection: Use an existing connection
   - DB Version: Neo4j 1.X.X
   - Server URL: "http://localhost:7474/db/data"

2. As in the tNeo4jOutput component, specify the URL of the Neo4j server to connect to, "http://localhost:7474/db/data" in this example.

3. Click the [...] button next to Edit schema and define the schema for employees information display. When done, click OK to close the [Schema] dialog box and propagate the schema to the next component.

   ![Schema dialog box](image)

   - The defined schema columns automatically appear in the Mapping table.
4. In the **Query** field, type in the Cypher query to match the data to read from the Neo4j database. In this example, use the following Cypher query to find employees who are more than 40 years old and are under the manager *m6*.

   "MATCH (n) WHERE n.age > 40 AND n.managerID = 'm6' RETURN n;"

5. Fill the **Return parameter** field for each schema column with a return parameter in double quotes to map the node properties in the Neo4j database with the schema columns.

6. Double-click the **tLogRow** component to open its **Basic settings** view, and select the **Table (print values in cells of a table)** option to display the retrieved information in a table.

### Executing the Job

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** or click **Run** on the **Run** tab to run the Job.

The employees data of the CSV file is written to the Neo4j database and then the information of employees matching the set conditions is retrieved from the Neo4j database and displayed on the console.

### Scenario 2: Writing family information to Neo4j and creating relationships

This scenario applies only to a Talend solution with Big Data.

This scenario describes a Job that will write family information to labeled nodes in a remote Neo4j database and create relationships based on the family names.

### Adding and linking components

**Procedure**

1. Create a Job and add the following components to the Job by typing theirs names in the design workspace or dropping them from the **Palette**:
• a **tFileInputDelimited** component, to read the family data from a CSV file,

• a **tNeo4jOutput** component to write the family data to a Neo4j database and create relationships between husband and wife.

2. Link the **tFileInputDelimited** component to the **tNeo4jOutput** component using a **Row > Main** connection.

3. Label the components to better identify their roles in the Job.

![Diagram showing the connection between tFileInputDelimited and tNeo4jOutput components]

**Configuring the components**

**Configuring the data source**

**Procedure**

1. Double-click the **tFileInputDelimited** component to open its **Basic settings** view on the **Components** tab.

![Screenshot of the tFileInputDelimited component settings]

2. In the **File name/Stream** field, specify the path to the CSV file that contains the family data to read. The input CSV file used in this example is as follows:

   Name;Gender;Age;Family
   Jenny;Female;24;the Johnsons
   Jack;Male;26;the Johnsons
   Richard;Male;35;the Blacks
   Anne;Female;36;the Whites
   Helen;Female;28;the Blacks
   Tom;Male;38;the Whites

3. In the **Header** field, specify the number of rows to skip as header rows. In this example, the first row of the CSV file is the header row.

4. Click the [...] button next to **Edit schema** to open the [Schema] dialog box, and define the input schema based on the structure of the input file. In this example, the input schema is composed of six columns: *name* (integer), *gender* (String), *age* (Integer), and *family* (String).
When done, click **OK** to close the [Schema] dialog box and propagate the schema to the next component.

Writing data to Neo4j and creating indexes and relationships

Procedure

1. Click the **tNeo4jOutput** component and select the **Component** tab to open its **Basic settings** view.

2. From the **DB Version** list, select **Neo4J 2.X.X** to enable node labeling.

3. Define a Neo4j database connection. In this example, the Neo4j database is accessible in REST mode, so select the **Remote server** check box and specify the URL of the Neo4j server in the **Server URL** field, "http://localhost:7474/db/data" in this example.
4. Double-click the tNeo4jOutput component or click the Mapping button on the component’s Basic settings view to open the index and relationship mapping editor.

5. With the name column selected from the schema panel, click the Index creation tab, click the [+] button to add a row in the table, and create an index named first_name on this column:
   - In the Name field, enter first_name between double quotation marks.
   - In the Key field, enter first_name between double quotation marks to give the index a key.
   Then click in the schema panel to validate your index creation.

6. With the family column selected from the schema panel, click the Index creation tab, click the [+] button to add a row in the table, and create an index named family on this column:
   - In the Name field, enter family between double quotation marks.
   - In the Key field, enter family_name between double quotation marks to give the index a key.
   Then click in the schema panel to validate your index creation.

7. With the family column selected from the schema panel, click the Relationship creation tab, click the [+] button to add a row in the table, and create a relationship named Spouse on this column based on the index named family:
   - In the Type field, enter Spouse between double quotation marks.
   - From the Direction list field, select either Outgoing or Incoming.
   - In the Index Name field, enter family between double quotation marks.
   - In the Index Key field, enter family_name between double quotation marks.
Then click in the schema panel to validate your relationship creation, and click OK to close the mapping editor.

8. Select the Use label (Neo4j > 2.0) check box and enter Families between double quotation marks in the Label name field so that the nodes to be created will be labeled Families.

9. From the Data action list, select Insert or update, and set a reference key in the Index area that appears:
   - In Index name field, enter first_name between double quotation marks.
   - In Index key field, enter first_name between double quotation marks.
   - From Index value field, select name. As the Value field is left blank in index creation, the index value will be the value of the name column for each row.

   This way, when the Job is executed, nodes will be inserted or updated in the Neo4j database based on the first_name index: for each data row, if a node containing the same first name already exists in the database, the node will be updated; otherwise, a new node will be created.

**Executing the Job and checking the result**

**Procedure**

1. Press Ctrl+S to save the Job, and press F6 or click Run on the Run tab to run the Job.
2. In the address bar of your Web browser, enter the URL of the Neo4j database browser, `http://localhost:7474/` in this example, and enter the following Cypher query in the command line to view the nodes.

MATCH (n:`Families`) RETURN n;

As shown in the graphic view, three pairs of nodes labeled *Families* have been created and those with the same family name are linked together via the relationship *Spouse*. 
tNeo4jOutputRelationship

Receives data from the preceding component and writes relationships into Neo4j.

**tNeo4jOutputRelationship Standard properties**

These properties are used to configure tNeo4jOutputRelationship running in the Standard Job framework.

The Standard tNeo4jOutputRelationship component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Use existing connection</strong></th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB version</strong></td>
<td>Select the Neo4j version you are using. This list is not shown if the Use an existing connection check box is selected. Upon selecting a database version, you will be prompted to install the corresponding database driver JAR files if not yet installed. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
<tr>
<td><strong>Remote server</strong></td>
<td>Select this check box if you use a Neo4j remote server, and specify the root URL in the Server URL field.</td>
</tr>
<tr>
<td><strong>Database path</strong></td>
<td>If you use Neo4j in embedded mode, specify the path of the data file. This field appears only if you do not select the Use an existing connection check box.</td>
</tr>
</tbody>
</table>
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  • View schema: choose this option to view the schema only.  
  • Change to built-in property: choose this option to change the schema to Built-in for local changes.  
  • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the...
Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

<table>
<thead>
<tr>
<th>Server url</th>
<th>If you use Neo4j in REST mode, specify the root URL. This field appears only if you select the Remote server check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shutdown after job</strong></td>
<td>Select this check box to shutdown the Neo4j database connection when no more operations on Neo4j are going to be performed after the current component. Alternatively, you can use tNeo4jClose to shutdown the database. This avoids errors such as &quot;Id file not properly shutdown&quot; at next execution of Jobs involving Neo4j. This check box is available only if the Use an existing connection check box is not selected.</td>
</tr>
<tr>
<td><strong>Relationship type</strong></td>
<td>Specify the relationship type to create for each row.</td>
</tr>
<tr>
<td><strong>Start node / Index name</strong></td>
<td>Specify the index name to query the starting node of the newest relationship.</td>
</tr>
<tr>
<td><strong>Start node / Index key</strong></td>
<td>Specify the index key to query the starting node.</td>
</tr>
<tr>
<td><strong>Start node / Index value</strong></td>
<td>Select the index value to query the starting node.</td>
</tr>
</tbody>
</table>
| **Relationship direction** | Select a relationship direction.  
  - **Outgoing**: The relationship starts from the start node to the end node.  
  - **Incoming**: The relationship starts from the end node to the start node. |
| **End node / Index name** | Specify the index name to query the ending node of the newest relationship. |
| **End node / Index key** | Specify the index key to query the ending node. |
End node / Index value | Select the index value to query the ending node.
---|---
Mapping | Use this table to map relationship properties with the input schema columns.

**Advanced settings**

**Commit every** | Enter the number of rows to be completed before committing batches of nodes together the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.
| This option is only supported by the embedded mode of the database. You can't make transactions in REST mode.

**Batch import** | Select this check to activate the batch mode
| This option is only supported by the embedded mode of the database. **Be very careful with the batch import mode. Because this option is unsafe, perform a backup before Job execution.**
| If you want more explanations about memory mapping configuration of batch import, please refer to Neo4j documentation at: [http://neo4j.com/docs/stable/batchinsert-examples.html](http://neo4j.com/docs/stable/batchinsert-examples.html)

**Node store mapped memory** | Type in the memory size in MB allocated to nodes.

**Relationship store mapped memory** | Type in the memory size in MB allocated to relationships.

**Property store mapped memory** | Type in the memory size in MB allocated to property.

**String store mapped memory** | Type in the memory size in MB in Mo allocated to strings.

**Array store mapped memory** | Type in the memory size in MB allocated to arrays.

**tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.
| **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.
| A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as an output component and it always needs an incoming link.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Scenario: Writing information of actors and movies to Neo4j with hierarchical relationship

This scenario applies only to a Talend solution with Big Data.

This scenario describes a Job that will import information about actors and movies from two CSV files into a remote Neo4j database and then create relationship between the actors and the movies based on another CSV file that describes the actors’ roles in the movies.

Adding and linking components

Procedure

1. Create a Job and add the following components to the Job by typing theirs names in the design workspace or dropping them from the Palette:
   - a tNeo4jConnection component, to open a connection to a Neo4j database,
   - three tFileInputDelimited components, to read the input information of actors and movies,
   - two tNeo4jOutput components, to write information of movies and actors to the connected Neo4j database,
   - a tNeo4jOutputRelationship component, to create relationship between actors and movies, and
   - a tNeo4jClose component, to close the previous opened Neo4j connection at the end of Job execution.

2. Link the first tFileInputDelimited to the first tNeo4jOutput component using a Row > Main connection. This sub-job will import the actors data into the Neo4j database.

3. Link the second tFileInputDelimited component to the second tNeo4jOutput component using a Row > Main connection. This sub-job will import the movies data into the Neo4j database.

4. Link the third tFileInputDelimited component to the tNeo4jOutputRelationship component using a Row > Main connection. This sub-job will create relationship between actors and movies.

5. Link the sub-jobs using Trigger > On Subjob Ok connections.

6. Label the components to better identify their roles in the Job.
Configuring the components

Opening a Neo4j database connection

Procedure

1. Double-click the tNeo4jConnection component to open its Basic settings view on the Component tab.
2. From the DB Version list, select Neo4J 2.X.X to enable node labeling.
3. Select the **Use a remote server** check box and specify the URL of the Neo4j server in the Server URL field, "http://localhost:7474/db/data" in this example.

### Importing the actors data into Neo4j

#### Procedure

1. Double-click the first tFileInputDelimited component to open its Basic settings view on the Component tab.

2. In the File name/Stream field, specify the path to the CSV file that describes the actors' names, the years they were born in, and their roles in the movies.

   The input CSV file used in this example is as follows:
   
<table>
<thead>
<tr>
<th>Name</th>
<th>Born</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keanu Reeves</td>
<td>1964</td>
</tr>
<tr>
<td>Laurence Fishburne</td>
<td>1961</td>
</tr>
<tr>
<td>Carrie-Anne Moss</td>
<td>1967</td>
</tr>
</tbody>
</table>

3. In the **Header** field, specify the number of rows to skip as header rows. In this example, the first row of the CSV file is the header row.

4. Click the [...] button next to Edit schema to open the [Schema] dialog box, and define the input schema based on the structure of the input file. In this example, the input schema is composed of two columns, **name** and **born**, both of type String.

   When done, click **OK** to close the [Schema] dialog box and propagate the schema to the next component.
5. Click the first **tNeo4jOutput** component and select the **Component** tab to open its **Basic settings** view.

6. Select the **Use an existing connection** check box to reuse the Neo4j database connection opened by the **tNeo4jConnection** component, which is the only connection component used in this example.

7. Double-click the **tNeo4jOutput** component or click the **Mapping** button on the component’s **Basic settings** view to open the index and relationship mapping editor.

8. With the **name** column selected from the schema panel, click the **Index creation** tab, click the [+] button to add a row in the table, and create an index named **name** on this column:
   - In the **Name** field, enter *name* between double quotation marks.
   - In the **Key** field, enter *name* between double quotation marks to give the index a key *name*.

Then click in the schema panel to validate your index creation, and click **OK** to close the mapping editor.
9. Select the **Use label (Neo4j > 2.0)** check box and enter *Actors* between double quotation marks in the **Label name** field so that the nodes to be created will be labeled *Actors*.

**Importing the movies data into Neo4j**

**Procedure**

1. Double-click the second **tFileInputDelimited** component to open its **Basic settings** view on the **Component** tab.

2. In the **File name/Stream** field, specify the path to the CSV file that contains the information of the movies.

   The input CSV file used in this example is as follows:

   ```
   Title;Released;Tagline
   ```
3. In the Header field, specify the number of rows to skip as header rows. In this example, the first row of the CSV file is the header row.

4. Click the [...] button next to Edit schema to open the [Schema] dialog box, and define the input schema based on the structure of the input file. In this example, the input schema is composed of three columns, title (type String), released (type Date), and tagline (type String). When done, click OK to close the [Schema] dialog box and propagate the schema to the next component.

5. Click the second tNeo4jOutput component and select the Component tab to open its Basic settings view.

6. Select the Use an existing connection check box to reuse the Neo4j database connection opened by the tNeo4jConnection component, which is the only connection component used in this example.

7. Double-click the tNeo4jOutput component or click the Mapping button on the component's Basic settings view to open the index and relationship mapping editor.

8. With the title column selected from the schema panel, click the Index creation tab, click the [+] button to add a row in the table, and create an index named title on this column:
   - In the Name field, enter title between double quotation marks.
   - In the Key field, enter title between double quotation marks to give the index a key.

Then click in the schema panel to validate your index creation, and click OK to close the mapping editor.
9. Select the **Use label (Neo4j > 2.0)** check box and enter *Movies* between double quotation marks in the **Label name** field so that the nodes to be created will be labeled *Movies*.

**Creating actor-movie relationship in Neo4j**

**Procedure**

1. Double-click the third **tFileInputDelimited** component to open its **Basic settings** view on the **Component** tab.

2. In the **File name/Stream** field, specify the path to the CSV file that describes the actors' roles in the movies.

The input CSV file used in this example is as follows:

```
Actor;Movie;Role
Keanu Reeves;The Matrix;Neo
```
3. In the **Header** field, specify the number of rows to skip as header rows. In this example, the first row of the CSV file is the header row.

4. Click the [...] button next to **Edit schema** to open the [Schema] dialog box, and define the input schema based on the structure of the input file. In this example, the input schema is composed of three columns, **actor**, **movie**, and **role**, all of type String.

When done, click **OK** to close the [Schema] dialog box and propagate the schema to the next component.

5. Double-click the **tNeo4jOutputRelationship** component to open its **Basic settings** view.

6. Select the **Use an existing connection** check box to reuse the Neo4j database connection opened by the **tNeo4jConnection** component, which is the only connection component used in this example.

7. In the **Relationship type** field, enter the type of relationship to be created, between double quotation marks. In this example, the relationship type is **ACTED_IN**, which will appear as the label of the relationship between nodes.

8. Define the start and end nodes and the direction of the relationship to be created. In this example, the relationship will be created from **Actors** nodes, on which an index called **name** has been defined, to **Movies** nodes, on which an index called **title** has been defined.

   - **Start node**: enter **name** between double quotation marks in the **Index name** and **Index key** fields, and select the schema column **name** from the **Index value** list. As the **Value** field is left blank in index creation, the index value will be the value of column **name** for each row.
• From the **Relationship direction** list, select **Outgoing**.
• **End node**: enter *title* between double quotation marks in the **Index name** and **Index key** fields, and select the schema column *title* from the **Index value** list. As the **Value** field is left blank in index creation, the index value will be the value of column *title* for each row.

9. Click the [+ ] button three times to add three rows in the **Mapping** table to define three properties of the relationship:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value taken from...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>actor column of the schema</td>
</tr>
<tr>
<td>Movie</td>
<td>movie column of the schema</td>
</tr>
<tr>
<td>Role</td>
<td>role column of the schema</td>
</tr>
</tbody>
</table>

**Executing the Job and checking the result**

**Procedure**

1. Press **Ctrl+S** to save the Job, and press **F6** or click **Run** on the **Run** tab to run the Job.
2. In the address bar of your Web browser, enter the URL of the Neo4j database browser, *http://localhost:7474/* in this example, and enter the following Cypher query in the command line to view the nodes with the specified relationship between them.

   ```cypher
   MATCH (a)-[:`ACTED_IN`]->(b) RETURN a,b;
   ```

   As shown in the graphic view, *Actors* nodes are linked with *Movies* nodes via the relationship *ACTED_IN*, with properties as defined in the Job.
tNeo4jRow

Executes the stated Cypher query onto the specified Neo4j database.

Depending on the nature of the query, tNeo4jRow acts on the data (although without handling data).
tNeo4jRow is the specific component for Neo4J database query. It implements a flow in the Job design although it doesn't provide output.

**tNeo4jRow Standard properties**

These properties are used to configure tNeo4jRow running in the Standard Job framework.

The Standard tNeo4jRow component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Use an existing connection</strong></th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB version</strong></td>
<td>Select the Neo4j version you are using.</td>
</tr>
<tr>
<td></td>
<td>This list is not shown if the <strong>Use an existing connection</strong> check box is selected.</td>
</tr>
<tr>
<td></td>
<td>Upon selecting a database version, you will be prompted to install the corresponding database driver JAR files if not yet installed. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
<tr>
<td><strong>Remote server</strong></td>
<td>Select this check box if you use a Neo4j remote server, and specify the root URL in the <strong>Server URL</strong> field.</td>
</tr>
<tr>
<td><strong>Database path</strong></td>
<td>If you use Neo4j in embedded mode, specify the directory to hold your data files. If the specified directory does not exist, it will be created.</td>
</tr>
<tr>
<td></td>
<td>This field appears only if you do not select the <strong>Use an existing connection</strong> check box or the <strong>Remote server</strong> check box.</td>
</tr>
<tr>
<td><strong>Shutdown after job</strong></td>
<td>Select this check box to shutdown the Neo4j database connection when no more operations on Neo4j are going to be performed after the current component.</td>
</tr>
<tr>
<td></td>
<td>Alternatively, you can use tNeo4jClose to shutdown the database.</td>
</tr>
<tr>
<td></td>
<td>This avoids errors such as “Id file not properly shutdown” at next execution of Jobs involving Neo4j.</td>
</tr>
<tr>
<td></td>
<td>This check box is available only if the <strong>Use an existing connection</strong> check box is not selected.</td>
</tr>
</tbody>
</table>
**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

**Query**
Enter your Cypher query in double quotes. If you have some parameters corresponding to the input schema columns, declare them with curly brackets ({}) and map them in the **Parameters** table.

**Parameters**
Click the `[+]` button to add parameters corresponding to your Cypher query and map them with the input schema columns.

- **Parameter name**: Type in the name of your parameter, in double quotes.
- **Parameter value**: Select the schema column you want to map your parameter with.

**Die on error**
This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows.
### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit every</td>
<td>Enter the number of rows to be completed before committing batches of nodes together the database. This option ensures transaction quality (but not rollback) and, above all, better performance at execution. This option is only supported by the embedded mode of the database. You cannot make transactions in REST mode, that is to say, not on a remote server in the deprecated 2.X.X versions.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_NODE_INSERTED: the number of nodes inserted. This is an After variable and it returns a long.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_RELATIONSHIP_INSERTED: the number of relationships inserted. This is an After variable and it returns a long.</td>
</tr>
<tr>
<td></td>
<td>NB_PROPERTY_UPDATED: the number of properties updated. This is an After variable and it returns a long.</td>
</tr>
<tr>
<td></td>
<td>NB_NODE_DELETED: the number of nodes deleted. This is an After variable and it returns a long.</td>
</tr>
<tr>
<td></td>
<td>NB_RELATIONSHIP_DELETED: the number of relationships deleted. This is an After variable and it returns a long.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as a standalone component or as an output component in a process.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Scenario 1: Creating nodes with a label using a Cypher query

This scenario applies only to a Talend solution with Big Data.

This scenario describes a Job that creates nodes with a label in an embedded Neo4j database using a Cypher query, and then displays some properties of the specified node on the console.

Adding and linking components

Procedure

1. Create a Job and add the following components to the Job by typing theirs names in the design workspace or dropping them from the Palette:
   - a `tNeo4jConnection` component, to open a connection to a Neo4j database,
   - a `tNeo4jRow` component, to create nodes in the Neo4j database with a Cypher query,
   - a `tNeo4jIntput` component, to read the specified properties of the created nodes,
   - a `tLogRow` component, to display the data on the Run console, and
   - a `tNeo4jClose` component, to close the Neo4j database connection established by the `tNeo4jConnection` component.

2. Link the `tNeo4jConnection` component to the `tNeo4jRow` component using a Trigger > On Subjob Ok connection.

3. Link the `tNeo4jRow` component to the `tNeo4jIntput` component using a Trigger > On Subjob Ok connection.

4. Link the `tNeo4jIntput` component to the `tLogRow` component using a Row > Main connection.

5. Link the `tNeo4jIntput` component to the `tNeo4jClose` component using a Trigger > On Subjob Ok connection.
Configuring the components

Configuring a Neo4j database connection and nodes creation

Procedure

1. Double-click the tNeo4jConnection component to open its Basic settings view on the Component tab.

2. From the DB Version list, select Neo4j 2.X.X to enable node labeling support.

3. In the Database path field, specify the directory for your data files, “E:/Talend/DB/Neo4j/data” in this example.

In this example, you will use Neo4j in embedded mode; to connect to a remote Neo4j server via REST, click the Use a remote server check box and specify the URL of the Neo4j server.
4. Double-click the **tNeo4jRow** component to open its **Basic settings** view on the **Component** tab.

5. Select the **Use an existing connection** check box to reuse the Neo4j database connection opened by the **tNeo4jConnection** component, which is the only connection component used in this example.

6. In the **Query** field, type in the Cypher query to be executed by the component.

   In this example, type in the following query to create two nodes with the label *Managers* and three properties - *id*, *name*, and *position*:

   ```cypher
   CREATE (a:Managers { id: 1, name: 'Gerald White', position: 'HR directory'})
   CREATE (b:Managers { id: 2, name: 'Jimmy Black', position: 'Sales manager'})
   ```

   As this component does not have an input flow, leave the schema and the parameter settings blank.

Configuring data retrieving and display

**Procedure**

1. Double-click the **tNeo4jInput** component to open its **Basic settings** view.

2. Select the **Use an existing connection** check box to reuse the connection opened by the **tNeo4jConnection** component.
3. Click the [...] button next to **Edit schema** and define the schema corresponding to the node properties you want to retrieve and display. In this example, the schema is composed of two columns, *name* and *position*, both of type String.

When done, click **OK** to close the [Schema] dialog box and propagate the schema to the next component.

![Schema of tNeo4jInput_1](image)

The defined schema columns automatically appear in the **Mapping** table.

4. In the **Query** field, type in the Cypher query to match the data to read from the Neo4j database. In this example, use the following Cypher query to find the name and position of the manager whose ID is 2.

   ```cypher
   "MATCH (ee:Managers{id:2}) RETURN ee.name, ee.position;"
   ```

   or

   ```cypher
   "MATCH (ee:Managers) WHERE ee.id = 2 RETURN ee.name, ee.position;"
   ```

5. Fill the **Return parameter** field for each schema column with a return parameter in double quotes to map the node properties in the Neo4j database with the schema columns.

6. Double-click the **tLogRow** component to open its **Basic settings** view, and select the **Table (print values in cells of a table)** option to display the retrieved information in a table.

**Executing the Job**

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** or click **Run** on the **Run** tab to run the Job.
Two nodes are created in the Neo4j database based on the Cypher query, with the data files written in the specified directory, and the defined information is retrieved and displayed on the console.

Scenario 2: Importing data from a CSV file to Neo4j using a Cypher query

This scenario applies only to a Talend solution with Big Data.

This scenario describes a Job that first imports employees data from a CSV file into a Neo4j database using a Cypher query, and then displays the information on the console.

Adding and linking components

Procedure

1. Create a Job and add the following components to the Job by typing theirs names in the design workspace or dropping them from the Palette:
   - a **tNeo4jConnection** component, to open a connection to a Neo4j database,
   - a **tFileInputDelimited** component, to read the source data from a CSV file,
   - a **tNeo4jRow** component, to write the employees data to the Neo4j database with a Cypher query,
   - a **tNeo4jIntput** component, to read the employees data from the Neo4j database,
   - a **tLogRow** component, to display the data on the Run console, and
   - a **tNeo4jClose** component, to close the Neo4j database connection opened by the **tNeo4jConnection** component.
2. Link the **tNeo4jConnection** component to the **tFileInputDelimited** component using a **Trigger > On Subjob Ok** connection.
3. Link the **tFileInputDelimited** component to the **tNeo4jRow** component using a **Row > Main** connection.
4. Link the **tFileInputDelimited** component to the **tNeo4jIntput** component using a **Trigger > On Subjob Ok** connection.
5. Link the **tNeo4jIntput** component to the **tLogRow** component using a **Row > Main** connection.
6. Link the **tNeo4jIntput** component to the **tNeo4jClose** component using a **Trigger > On Subjob Ok** connection.
7. Label the components to better identify their roles in the Job.
Configuring the components

Configuring a Neo4j database connection

Procedure

1. Double-click the tNeo4jConnection component to open its Basic settings view on the Component tab.

2. From the DB Version list, select Neo4j 2.X.X.

3. Select the Use a remote server check box and specify the URL of the Neo4j server in the Server URL field, “http://localhost:7474/db/data” in this example.
In this example, you will use Neo4j in REST mode; to connect to a remote Neo4j server in embedded mode, clear the **Use a remote server** check box and specify the Neo4j data file directory in the **Database path** field.

### Configuring data import

**Procedure**

1. Double-click the **tFileInputDelimited** component to open its **Basic settings** view on the **Component** tab.

2. **Input employees_info(tFileInputDelimited_1)**

   ![Input employees_info(tFileInputDelimited_1)](image)

   - **Property Type**: Built-in
   - **File name/Stream**: "E:/Telend/Data/Input/employees_info.csv"
   - **Row Separator**: ";"  
   - **Field Separator**: ";"
   - **Header**: 1  
   - **Limit**: 
   - **Schema**: 
     - **Skip empty rows**: 
     - **Uncompress as zip file**: 
     - **Die on error**: 

3. In the **File name/Stream** field, specify the path to the CSV file that contains the employees data to read.

   The input CSV file used in this example is as follows:

   ```
   employeeID;employeeName;age;hireDate;salary;managerID
   1;Rutherford Roosevelt;38;06-10-2008;13336.58;m5
   2;Warren Adams;43;05-22-2008;11626.68;m6
   3;Andrew Roosevelt;55;04-01-2007;10052.95;m4
   4;Herbert Quincy;54;06-14-2007;10694.71;m6
   5;Woodrow Polk;33;08-14-2007;13751.50;m4
   6;Theodore Johnson;47;01-26-2008;12426.87;m6
   7;Benjamin Adams;32;02-25-2008;10438.65;m4
   8;Woodrow Harrison;51;10-11-2008;11188.27;m5
   9;George Truman;40;04-28-2008;14254.49;m5
   10;Harry Jackson;38;04-01-2008;12798.78;m6
   ```

4. **In the **Header** field, specify the number of rows to skip as header rows. In this example, the first row of the CSV file is the header row.**

4. **Click the [...] button next to **Edit schema** to open the [Schema] dialog box, and define the input schema based on the structure of the input file. In this example, the input schema is composed of six columns: employeeID (integer), employeeName (String), age (Integer), hireDate (Date), salary (Double), and managerID (String).**

   When done, click **OK** to close the [Schema] dialog box and propagate the schema to the next component.
5. Double-click the **tNeo4jRow** component to open its **Basic settings** view on the **Component** tab.

6. Select the **Use an existing connection** check box to reuse the Neo4j database connection opened by the **tNeo4jConnection** component, which is the only connection component used in this example.

7. In the **Query** field, type in the Cypher query to be executed by the component.
   In this example, type in the following query to create nodes with the label **Employees** and six properties, to hold the data from the input flow:
   - **ID**, which will take the value of the variable parameter **id**,
   - **Name**, which will take the value of the variable parameter **name**,
• **Age**, which will take the value of the variable parameter *age*,
• **HireDate**, which will take the value of the variable parameter *hire_date*,
• **Salary**, which will take the value of the variable parameter *salary*, and
• **ManagerID**, which will take the value of the variable parameter *manager_id*.

```
"CREATE (n:Employees{ID:{id}, Name:{name}, Age:{age}, HireDate:{hire_date}, Salary:{salary}, ManagerID:{manager_id}})"
```

8. In the **Parameters** table, type in the variable parameters in the **Parameter** field in accordance with your Cypher query, and map each of them with an input schema column by selecting it from the **Parameter value** list field.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Parameter value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;id&quot;</td>
<td>employeeID</td>
</tr>
<tr>
<td>&quot;name&quot;</td>
<td>employeeName</td>
</tr>
<tr>
<td>&quot;age&quot;</td>
<td>age</td>
</tr>
<tr>
<td>&quot;hire_date&quot;</td>
<td>hireDate</td>
</tr>
<tr>
<td>&quot;salary&quot;</td>
<td>salary</td>
</tr>
<tr>
<td>&quot;manager_id&quot;</td>
<td>managerID</td>
</tr>
</tbody>
</table>

**Configuring data retrieving and display**

**Procedure**

1. Double-click the **tNeo4jInput** component to open its **Basic settings** view.

2. Select the **Use an existing connection** check box to reuse the connection opened the **tNeo4jConnection** component.
3. Click the [...] button next to **Edit schema** and define the schema corresponding to the node properties you want to retrieve and display.
When done, click **OK** to close the [Schema] dialog box and propagate the schema to the next component.

![Schema of Read_nodes_Employees](image)

The defined schema columns automatically appear in the **Mapping** table.

4. In the **Query** field, type in the Cypher query to match the data to read from the Neo4j database. In this example, use the following Cypher query to retrieve all the properties of all the nodes with the label **Employees**.

   "MATCH (n:Employees) RETURN *;"

5. Fill the **Return parameter** field for each schema column with a return parameter in double quotes to map the node properties in the Neo4j database with the schema columns.

6. Double-click the **tLogRow** component to open its **Basic settings** view, and select the **Table (print values in cells of a table)** option to display the retrieved information in a table.

**Executing the Job**

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** or click **Run** on the **Run** tab to run the Job.
The employees data in the CSV file is written to the Neo4j database and displayed on the console.

### Scenario 3: Importing data from a CSV file to Neo4j and creating relationships using a single Cypher query

This scenario applies only to a Talend solution with Big Data.

This scenario describes a Job that imports family information from a CSV file into a remote Neo4j database and create relationships between persons and families using a single Cypher query through a tNeo4jRow component.

Below is content of the CSV file to import data from in this example:

```
Name;Gender;Age;Family
Jenny;Female;24;the Johnsons
Jack;Male;26;the Johnsons
Richard;Male;35;the Blacks
Anne;Female;36;the Whites
Helen;Female;28;the Blacks
Tom;Male;38;the Whites
```

As `MERGE` is used with `LOAD CSV` in this example, to ensure the Cypher query is executed in an efficient way, another tNeo4jRow component is used to create an index on the property to merge.

### Adding and linking components

**Procedure**

1. Create a Job and add two tNeo4jRow components to the Job by typing the component name in the design workspace or dropping them from the Palette.
2. Link the components using a Trigger > On Sub Job Ok connection.
3. Label the components to better identify their roles in the Job.
Configuring the components

Creating an index

Procedure

1. Double-click the first tNeo4jRow component to open its Basic settings view on the Component tab.

2. From the DB Version list, select Neo4J 2.X.X.

3. Select the Remote server check box and specify the URL of the Neo4j server in the Server URL field, "http://localhost:7474/db/data" in this example.

4. In the Query field, type in the following query to create an index on the property you are going to merge, which is the name property of the Family nodes in this example:

   "CREATE INDEX ON :Family(name)"

Importing data and creating relationships

Procedure

1. Double-click the second tNeo4jRow component to open its Basic settings view on the Component tab.
2. From the **DB Version** list, select **Neo4J 2.X.X**.

3. Select the **Remote server** check box and specify the URL of the Neo4j server in the **Server URL** field, "http://localhost:7474/db/data" in this example.

4. In the **Query** field, type in the following Cypher query to import family data from the CSV file, create relevant **Person** and **Family** nodes, and create relationships between persons and families:

```
"LOAD CSV WITH HEADERS FROM 'file:E:/Talend/Data/Input/families.csv' AS csvLine 
FIELDTERMINATOR ';' 
MERGE (family:Family { name: csvLine.Family }) 
CREATE (person:Person { name: csvLine.Name, gender: csvLine.Gender, age: toInt(csvLine.Age)}) 
CREATE (person)-[:From]->(family)"
```

**Executing the Job and checking the result**

**Procedure**

1. Press **Ctrl+S** to save the Job, and press **F6** or click **Run** on the **Run** tab to run the Job.

2. In the address bar of your Web browser, enter the URL of the Neo4j database browser, http://localhost:7474/ in this example, and enter the following Cypher query in the command line to view the **Person** and **Family** nodes linked via the relationship **From**:

```
MATCH (a:Person)-[:`From`]->(b:Family) RETURN a,b;
```

As shown in the graphic view, nodes labeled **Family** and **Person** have been created and the nodes of persons from the same families are linked with the relevant **Family** nodes via the relationship **From**.
tNetezzaBulkExec

Offers gains in performance while carrying out the Insert operations to a Netezza database.

**tNetezzaBulkExec Standard properties**

These properties are used to configure tNetezzaBulkExec running in the Standard Job framework.

The Standard tNetezzaBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see **Talend Studio User Guide**.
<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time and that the table must exist for the insert operation to succeed.</td>
</tr>
<tr>
<td><strong>Schema</strong> and <strong>Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide. Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available: • View schema: choose this option to view the schema only. • Change to built-in property: choose this option to change the schema to Built-in for local changes. • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td><strong>File Name</strong></td>
<td>Name of the file to be loaded. This file should be located on the same machine where the Studio is installed or where the Job using tNetezzaBulkExec is deployed.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<p>| <strong>Field Separator</strong> | Character, string or regular expression to separate fields. |</p>
<table>
<thead>
<tr>
<th><strong>Escape character</strong></th>
<th>Character of the row to be escaped.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Date format / Date delimiter</strong></td>
<td>Use Date format to distinguish the way years, months and days are represented in a string. Use Date delimiter to specify the separator between date values.</td>
</tr>
<tr>
<td><strong>Time format / Time delimiter</strong></td>
<td>Use Time format to distinguish the time is represented in a string. Use Time delimiter to specify the separator between time values.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select the encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td><strong>Max Errors</strong></td>
<td>Enter the maximum error limit that will not stop the process.</td>
</tr>
<tr>
<td><strong>Skip Rows</strong></td>
<td>Enter the number of rows to be skipped.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Global Variables</strong></td>
</tr>
<tr>
<td><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Usage</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage rule</strong></td>
</tr>
<tr>
<td><strong>Dynamic settings</strong></td>
</tr>
</tbody>
</table>
when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

**Related scenarios**

For use cases in relation with **tNetezzaBulkExec**, see the following scenarios:

- *Scenario: Inserting transformed data in MySQL database* on page 2537.
- *Scenario: Inserting data in bulk in MySQL database* on page 2544.
- *Scenario: Truncating and inserting file data into an Oracle database* on page 2752.
tNetezzaClose

Closes the transaction committed in the connected Netezza database.

tNetezzaClose Standard properties

These properties are used to configure tNetezzaClose running in the Standard Job framework.
The Standard tNetezzaClose component belongs to the Databases family.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tNetezzaConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStat Catcher Statistics  | Select this check box to collect log data at the component level. |

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with Netezza components, especially with tNetezzaConnection and tNetezzaCommit.</th>
</tr>
</thead>
</table>
| Dynamic settings           | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.
Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.
For examples on using dynamic parameters, see Scenario: Reading data from databases through context- |
Based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

No scenario is available for the Standard version of this component yet.
tNetezzaCommit

Validates the data processed through the Job into the connected Netezza database.

Using a unique connection, tNetezzaCommit commits in one go a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

**tNetezzaCommit Standard properties**

These properties are used to configure tNetezzaCommit running in the Standard Job framework.

The Standard tNetezzaCommit component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tNetezzaConnection</strong> component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Warning:**

*If you want to use a Row > Main connection to link tNetezzaCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.*

### Advanced settings

<table>
<thead>
<tr>
<th>tStat Catcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other tNetezza* components, especially with the tNetezzaConnection and tNetezzaRollback components.</th>
</tr>
</thead>
</table>
Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenario

For tNetezzaCommit related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476.
tNetezzaConnection

Opens a connection to a Netezza database to be reused in the subsequent subjob or subjobs.

**tNetezzaConnection Standard properties**

These properties are used to configure tNetezzaConnection running in the Standard Job framework.
The Standard tNetezzaConnection component belongs to the Databases and the ELT families.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Additional JDBC Parameters</td>
<td>Specify additional connection properties for the DB connection you are creating.</td>
</tr>
<tr>
<td>Use or register a shared DB Connection</td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the Shared DB Connection Name field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.</td>
</tr>
</tbody>
</table>
This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the `tRunJob` component. Using a shared connection together with a `tRunJob` component with either of these two options enabled will cause your Job to fail.

### Advanced settings

| **Auto Commit** | Select this check box to commit any changes to the database automatically upon the transaction. With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component. Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component. |
| **tStatCatcher Statistics** | Select this check box to gather the job processing metadata at a Job level as well as at each component level. |

### Usage

| **Usage rule** | This component is more commonly used with other `tNetezza*` components, especially with the `tNetezzaCommit` and `tNetezzaRollback` components. |
| **Limitation** | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |

### Related scenarios

For a `tNetezzaConnection` related scenario, see [Scenario: Inserting data in mother/daughter tables](#) on page 2476.
tNetezzaInput

Reads a database and extracts fields from a Netezza database based on a query.

tNetezzaInput executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Main row link.

tNetezzaInput Standard properties

These properties are used to configure tNetezzaInput running in the Standard Job framework.
The Standard tNetezzaInput component belongs to the Databases family.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
</tbody>
</table>

  **Built-in**: No property data stored centrally.

  **Repository**: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.

Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.
For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

Use an existing connection

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

Note:
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component.
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Click Edit schema</strong></td>
<td>to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td><strong>Table Name</strong></td>
<td>Name of the table to be read.</td>
</tr>
<tr>
<td><strong>Query type and Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
</tbody>
</table>
Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use cursor</td>
<td>When selected, helps to decide the row set to work with at a time and thus optimize performance.</td>
</tr>
<tr>
<td>Trim all the String/Char columns</td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</td>
</tr>
<tr>
<td>Trim column</td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>QUERY</td>
<td>the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component covers all possible SQL queries for Netezza databases.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dynamic settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</td>
<td></td>
</tr>
</tbody>
</table>

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defin
ed, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the <strong>Install</strong> button on the <strong>Component</strong> tab view. You can also find out and add all missing JARs easily on the <strong>Modules</strong> tab in the <strong>Integration</strong> perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

**Related scenarios**

Related scenarios for **tNetezzaInput** are:

- **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489.
tNetezzaNzLoad

Inserts data into a Netezza database table using Netezza's nzload utility.

tNetezzaNzLoad bulk loads data into a Netezza table either from an existing data file, an input flow, or directly from a data flow in streaming mode through a named-pipe.

**tNetezzaNzLoad Standard properties**

These properties are used to configure tNetezzaNzLoad running in the Standard Job framework.
The Standard tNetezzaNzLoad component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property type</strong></td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of the DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the Netezza database.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table into which the data is to be inserted.</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td>On the table defined, you can perform one of the following operations before loading the data: None: No operation is carried out. Drop and create a table: The table is removed and created again. Create a table: The table does not exist and gets created. Create table if not exists: The table is created if it does not exist. Drop table if exists and create: The table is removed if it already exists and created again. Clear table: The table content is deleted before the data is loaded.</td>
</tr>
<tr>
<td><strong>Truncate table</strong>: executes a truncate statement prior to loading the data to clear the entire content of the table.</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td></td>
</tr>
<tr>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
<td></td>
</tr>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
<td></td>
</tr>
<tr>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td><strong>Data file</strong></td>
<td></td>
</tr>
<tr>
<td>Full path to the data file to be used. If this component is used on its own (not connected to another component with input flow) then this is the name of an existing data file to be loaded into the database. If it is connected, with an input flow to another component; this is the name of the file to be generated and written with the incoming data to later be used with nzload to load into the database.</td>
<td></td>
</tr>
<tr>
<td><strong>Use named-pipe</strong></td>
<td></td>
</tr>
<tr>
<td>Select this check box to use a named-pipe instead of a data file. This option can only be used when the component is connected with an input flow to another component. When the check box is selected, no data file is generated and the data is transferred to nzload through a named-pipe. This option greatly improves performance in both Linux and Windows.</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td></td>
</tr>
<tr>
<td>This component on named-pipe mode uses a JNI interface to create and write to a named-pipe on any Windows platform. Therefore the path to the associated JNI DLL must be configured inside the java library path. The component comes with two DLLs for both 32 and 64 bit operating systems that are...</td>
<td></td>
</tr>
<tr>
<td>Named-pipe name</td>
<td>Specify a name for the named-pipe to be used. Ensure that the name entered is valid.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| Use existing control file | Select this check box to provide a control file to be used with the nzload utility instead of specifying all the options explicitly in the component. When this check box is selected, **Data file** and the other nzload related options no longer apply. Please refer to Netezza's nzload manual for details on creating a control file. |

### Note:

The global variable `NB_LINE` is not supported when any control file is being used.

| Control file | Enter the path to the control file to be used, between double quotation marks, or click `[...]` and browse to the control file. This option is passed on to the nzload utility via the `-cf` argument. |

| Field separator | Character, string or regular expression used to separate fields. |

### Warning:

This is nzload's delim argument. If you do not use the **Wrap quotes around fields** option, you must make sure that the delimiter is not included in the data that's inserted to the database. The default value is `\t` or TAB. To improve performance, use the default value.

| Wrap quotes around fields | This option is only applied to columns of String, Byte, Byte[], Char, and Object types. Select either: |

#### None:

do not wrap column values in quotation marks.

#### Single quote:

wrap column values in single quotation marks.

#### Double quote:

wrap column values in double quotation marks.

### Warning:

If using the **Single quote** or **Double quote** option, it is necessary to use `\` as the **Escape char**.

| Advanced options | Set the nzload arguments in the corresponding table. Click `[+]` as many times as required to add arguments to the table. Click the **Parameter** field and choose among the arguments from the list. Then click the corresponding **Value** field and enter a value between quotation marks. |

For details about the available parameters, see **Parameters** on page 2691.
### Encoding
Select the encoding type from the list.

### Specify nzload path
Select this check box to specify the full path to the nzload executable. You must check this option if the nzload path is not specified in the PATH environment variable.

### Full path to nzload executable
Full path to the nzload executable on the machine in use. It is advisable to specify the nzload path in the PATH environment variable instead of selecting this option.

### tStatCatcher Statistics
Select this check box to collect log data at the component level.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is mainly used when non particular transformation is required on the data to be loaded on to the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This component can be used as a standalone or an output component.</td>
</tr>
</tbody>
</table>

### Parameters
The following table lists the parameters you can use in the Advanced options table in the Advanced settings tab.

<p>| -If | Name of the log file to generate. The logs will be appended if the log file already exists. If the parameter is not specified, the default name for the log file is <code>&lt;table_name&gt;.&lt;db_name&gt;.nzlog</code>. And it’s generated under the current working directory where the job is running. |</p>
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>-bf</strong></td>
<td>Name of the bad file to generate. The bad file contains all the records that could not be loaded due to an internal Netezza error. The records will be appended if the bad file already exists. If the parameter is not specified, the default name for the bad file is <code>&lt;table_name&gt;.&lt;db_name&gt;.nzbad</code>. And it’s generated under the current working directory where the job is running.</td>
</tr>
<tr>
<td><strong>-outputDir</strong></td>
<td>Directory path to where the log and the bad file are generated. If the parameter is not specified the files are generated under the current directory where the job is currently running.</td>
</tr>
<tr>
<td><strong>-logFileSize</strong></td>
<td>Maximum size for the log file. The value is in MB. The default value is 2000 or 2GB. To save hard disk space, specify a smaller amount if your job runs often.</td>
</tr>
<tr>
<td><strong>-compress</strong></td>
<td>Specify this option if the data file is compressed. Valid values are &quot;TRUE&quot; or &quot;FALSE&quot;. Default value if &quot;FALSE&quot;. This option is only valid if this component is used by itself and not connected to another component via an input flow.</td>
</tr>
<tr>
<td><strong>-skipRows &lt;n&gt;</strong></td>
<td>Number of rows to skip from the beginning of the data file. Set the value to ‘1’ if you like to skip the header row from the data file. The default value is ‘0’. This option should only be used if this component is used by itself and not connected to another component via an input flow.</td>
</tr>
<tr>
<td><strong>-maxRows &lt;n&gt;</strong></td>
<td>Maximum number of rows to load from the data file. This option should only be used if this component is used by itself and not connected to another component via an input flow.</td>
</tr>
<tr>
<td><strong>-maxErrors</strong></td>
<td>Maximum number of error records to allow before terminating the load process. The default value is ‘1’.</td>
</tr>
<tr>
<td><strong>-ignoreZero</strong></td>
<td>Binary zero bytes in the input data will generate errors. Set this option to ‘NO’ to generate error or to ‘YES’ to ignore zero bytes. The default value is ‘NO’.</td>
</tr>
<tr>
<td><strong>-requireQuotes</strong></td>
<td>This option requires all the values to be wrapped in quotes. The default value is &quot;FALSE&quot;. This option currently does not work with input flow. Use this option only in standalone mode with an existing file.</td>
</tr>
<tr>
<td><strong>-nullValue &lt;token&gt;</strong></td>
<td>Specify the token to indicate a null value in the data file. The default value is ‘NULL’. To improve slightly performance you can set this value to an empty field by specifying the value as single quotes: &quot;&quot;.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>fillRecord</td>
<td>Treat missing trailing input fields as null. You do not need to specify a value for this option in the value field of the table. This option is not turned on by default, therefore input fields must match exactly all the columns of the table by default. Trailing input fields must be nullable in the database.</td>
</tr>
<tr>
<td>ctrlChar</td>
<td>Accept control chars in char/varchar fields (must escape NUL, CR and LF). You do not need to specify a value for this option in the value field of the table. This option is turned off by default.</td>
</tr>
<tr>
<td>ctnString</td>
<td>Accept un-escaped CR in char/varchar fields (LF becomes only end of row). You do not need to specify a value for this option in the value field of the table. This option is turned off by default.</td>
</tr>
<tr>
<td>truncString</td>
<td>Truncate any string value that exceeds its declared char/varchar storage. You do not need to specify a value for this option in the value field of the table. This option is turned off by default.</td>
</tr>
<tr>
<td>dateStyle</td>
<td>Specify the date format in which the input data is written in. Valid values are: “YMD”, “Y2MD”, “DMY”, “DMY2”, “MDY”, “MDY2”, “MONDY”, “MONDY2”. The default value is “YMD”. The date format of the column in the component’s schema must match the value specified here. For example if you want to load a DATE column, specify the date format in the component schema as ’yyyy-MM-dd” and the -dateStyle option as ’YMD’. For more description on loading date and time fields, see Loading DATE, TIME and TIMESTAMP columns on page 2694.</td>
</tr>
<tr>
<td>dateDelim</td>
<td>Delimiter character between date parts. The default value is ”-“ for all date styles except for “MONDY[2]” which is ” ” (empty space). The date format of the column in the component’s schema must match the value specified here.</td>
</tr>
<tr>
<td>y2Base</td>
<td>First year expressible using two digit year (Y2) dateStyle.</td>
</tr>
<tr>
<td>timeStyle</td>
<td>Specify the time format in which the input data is written in. Valid values are: ”24HOUR” and ”12HOUR”. The default value is ”24HOUR”. For slightly better performance you should keep the default value. The time format of the column in the component’s schema must match the value specified here. For example if you want to load a TIME column, specify the date format in the component schema as ”HH:mm:ss” and the -timeStyle option as ”24HOUR”. For more description on loading date and time fields, see Loading DATE, TIME and TIMESTAMP columns on page 2694.</td>
</tr>
</tbody>
</table>
-timeDelim

Delimiter character between time parts. The default value is ".".

**Note:**
The time format of the column in the component’s schema must match the value specified here.

-timeRoundNanos

Allow but round non-zero digits with smaller than microsecond resolution.

-boolStyle

Specify the format in which Boolean data is written in the data. The valid values are: "1_0", "T_F", "Y_N", "TRUE_FALSE", "YES". The default value is "1_0". For slightly better performance keep the default value.

-allowRelay

Allow load to continue after one or more SPU reset or failed over. The default behaviour is not allowed.

-allowRelay <n>

Specify number of allowable continuation of a load. Default value is "1".

### Loading DATE, TIME and TIMESTAMP columns

When this component is used with an input flow, the date format specified inside the component’s schema must match the value specified for -dateStyle, -dateDelim, -timeStyle, and -timeDelim options.

<table>
<thead>
<tr>
<th>DB Type</th>
<th>Schema date format</th>
<th>-dateStyle</th>
<th>-dateDelim</th>
<th>-timeStyle</th>
<th>-timeDelim</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>&quot;yyyy-MM-dd&quot;</td>
<td>&quot;YMD&quot;</td>
<td>'-'</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>TIME</td>
<td>&quot;HH:mm:ss&quot;</td>
<td>n/a</td>
<td>n/a</td>
<td>&quot;24HOUR&quot;</td>
<td>':'</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>&quot;yyyy-MM-dd HH:mm:ss&quot;</td>
<td>&quot;YMD&quot;</td>
<td>'-'</td>
<td>&quot;24HOUR&quot;</td>
<td>':'</td>
</tr>
</tbody>
</table>

### Related scenario

For a related use case, see Scenario: Inserting data in bulk in MySQL database on page 2544.
tNetezzaOutput

 Writes, updates, makes changes or suppresses entries in a Netezza database.

 tNetezzaOutput executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the designed Job.

 tNetezzaOutput Standard properties

 These properties are used to configure tNetezzaOutput running in the Standard Job framework.

 The component in this framework is generally available.

 Note:

 This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

 For more information about dynamic database connectors, see Dynamic database components on page 598.

 Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

 ![Icon](image)

 Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.

 For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

 Use an existing connection |

 Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

 Note:

 When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

 1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component.
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td>Default: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td>Drop and create a table: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td>Create a table: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td>Create a table if not exists: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td>Drop a table if exists and create: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td>Clear a table: The table content is deleted.</td>
</tr>
<tr>
<td>Action on data</td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td>Insert: Add new entries to the table. If duplicates are found, job stops.</td>
</tr>
<tr>
<td></td>
<td>Update: Make changes to existing entries</td>
</tr>
<tr>
<td></td>
<td>Insert or update: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td>Update or insert: Update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
</tr>
<tr>
<td></td>
<td>Delete: Remove entries corresponding to the input flow.</td>
</tr>
<tr>
<td></td>
<td>Warning: It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking...</td>
</tr>
</tbody>
</table>
**Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the Update and Delete operations. To do that: Select the **Use field options** check box and then in the **Key in update column**, select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the **Key in delete column** for the Delete operation.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

**Built-In:** You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

**Repository:** You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Die on error

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.
## Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional JDBC parameters</strong></td>
<td>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <em>Use an existing connection</em> check box in the <strong>Basic settings</strong>.</td>
</tr>
<tr>
<td><strong>Use Batch</strong></td>
<td>Select this check box to activate the batch mode for data processing.</td>
</tr>
<tr>
<td><strong>Batch Size</strong></td>
<td>Specify the number of records to be processed in each batch. This field appears only when the <strong>Use batch mode</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Commit every</strong></td>
<td>Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at executions.</td>
</tr>
<tr>
<td><strong>Additional Columns</strong></td>
<td>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Type in the name of the schema column to be altered or inserted as new column.</td>
</tr>
<tr>
<td><strong>SQL expression</strong></td>
<td>Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td>Select <strong>Before</strong>, <strong>Replace</strong> or <strong>After</strong> following the action to be performed on the reference column.</td>
</tr>
<tr>
<td><strong>Reference column</strong></td>
<td>Type in a column of reference that the tDBOutput can use to place or replace the new or altered column.</td>
</tr>
<tr>
<td><strong>Use field options</strong></td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_LINE_UPDATED: the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_INSERTED: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_DELETED: the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_REJECTED: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a Netezza database. It also allows you to create a reject flow using a Row &gt; Rejects link to filter data in error. For an example of tMySqlOutput in use, see Scenario: Retrieving data in error with a Reject link on page 2529.</td>
</tr>
</tbody>
</table>

| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |
|-----------------| The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. |
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

**Related scenarios**

For **tNetezzaOutput** related topics, see:

- Scenario: Inserting a column and altering data using tMysqlOutput on page 2521.
tNetezzaRollback

Cancels the transaction committed in the connected Netezza database to avoid committing part of a transaction involuntarily.

### tNetezzaRollback Standard properties

These properties are used to configure tNetezzaRollback running in the Standard Job framework.

The Standard tNetezzaRollback component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

#### Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Component list</td>
<td>Select the <strong>tNetezzaConnection</strong> component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

#### Advanced settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

#### Usage

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage rule</td>
<td>This component is more commonly used with other tNetezza* components, especially with the <strong>tNetezzaConnection</strong> and <strong>tNetezzaCommit</strong> components.</td>
</tr>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</td>
</tr>
</tbody>
</table>
Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Related scenarios

For **tNetezzaRollback** related scenario, see **Scenario: Rollback from inserting data in mother/daughter tables** on page 2480.
tNetezzaRow

Executes the SQL query stated onto the specified Netezza database.

Depending on the nature of the query and the database, tNetezzaRow acts on the actual DB structure or on the data (although without handling data). The SQLBuilder tool helps you write easily your SQL statements.

tNetezzaRow is the specific component for this database query. It implements a flow in the job design although it does not provide output.

tNetezzaRow Standard properties

These properties are used to configure tNetezzaRow running in the Standard Job framework.

The Standard tNetezzaRow component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in:</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository:</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

Note:

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
</tbody>
</table>

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<p>| <strong>Table Name</strong> | Enter the name of the table to be processed. |
| <strong>Query type</strong> | Either <strong>Built-in</strong> or <strong>Repository</strong>. |
| <strong>Built-in</strong>: Fill in manually the query statement or build it graphically using SQLBuilder |</p>
<table>
<thead>
<tr>
<th>Repository:</th>
<th>Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row &gt; Rejects link.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **Additional JDBC parameters** | Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the Use an existing connection check box in the Basic settings. |
| **Propagate QUERY’s recordset** | Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the use column list. |
| | **Note:** |
| | This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of Object and this component is usually followed by tParseRecordSet. |
| **Use PreparedStatement** | Select this check box if you want to query the database using a PreparedStatement. In the Set PreparedStatement Parameter table, define the parameters represented by “?” in the SQL instruction of the Query field in the Basic Settings tab. |
| | **Parameter Index:** Enter the parameter position in the SQL instruction. |
| | **Parameter Type:** Enter the parameter type. |
| | **Parameter Value:** Enter the parameter value. |
| | **Note:** |
| | This option is very useful if you need to execute the same query several times. Performance levels are increased |
| **Commit every** | Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions. |
| **tStat Catcher Statistics** | Select this check box to collect log data at the component level. |
### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUERY</td>
<td>the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>NB_EFFECTED</td>
<td>the number of effected rows. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component offers the flexibility of the DB query and covers all possible SQL queries.</td>
<td></td>
</tr>
</tbody>
</table>

### Dynamic settings

<table>
<thead>
<tr>
<th>Dynamic settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</td>
<td></td>
</tr>
</tbody>
</table>

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided.

You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find...
Related scenarios

For related scenarios, see:

- Scenario: Combining two flows for selective output on page 2558
- Scenario: Removing and regenerating a MySQL table index on page 2552
tNetezzaSCD

Reflects and tracks changes in a dedicated Netezza SCD table.

tNetezzaSCD addresses Slowly Changing Dimension needs, reading regularly a source of data and logging the changes into a dedicated SCD table.

**tNetezzaSCD Standard properties**

These properties are used to configure tNetezzaSCD running in the Standard Job framework.

The Standard tNetezzaSCD component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the Repository file where properties are stored. The following fields are pre-filled in using fetched data.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td><strong>1.</strong> In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
<td></td>
</tr>
<tr>
<td><strong>2.</strong> In the child level, use a dedicated connection component to read that registered database connection.</td>
<td></td>
</tr>
</tbody>
</table>
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
</tbody>
</table>
| **Action on table** | Select to perform one of the following operations on the table defined:  
  **None**: No action is carried out on the table.  
  **Create table**: A new table is created.  
  **Create table if not exists**: A table is created if it does not exist. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  • **View schema**: choose this option to view the schema only.  
  • **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.  
  
  **Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.  
  
  **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.  
  
  **SCD Editor** | The SCD editor helps to build and configure the data flow for slowly changing dimension outputs. |
For more information, see SCD management methodology on page 2566.

<table>
<thead>
<tr>
<th>Use memory saving Mode</th>
<th>Select this check box to maximize system performance.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source keys include Null</td>
<td>Select this check box to allow the source key columns to have Null values.</td>
</tr>
<tr>
<td><strong>Warning:</strong></td>
<td>Special attention should be paid to the uniqueness of the source key(s) values when this option is selected.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Additional JDBC parameters</th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the Use an existing connection check box in the Basic settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>You can press <strong>Ctrl+Space</strong> to access a list of predefined global variables.</td>
</tr>
<tr>
<td>End date time details</td>
<td>Specify the time value of the SCD end date time setting in the format of <strong>HH:mm:ss</strong>. The default value for this field is <strong>12:00:00</strong>. This field appears only when SCD Type 2 is used and Fixed year value is selected for creating the SCD end date.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
<tr>
<td>Debug mode</td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE_UPDATED</strong>: the number of rows updated. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_REJECTED</strong>: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

| Usage rule | This component is used as an Output component. It requires an Input component and Row main link as input. |

| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

| Limitation | This component does not support using SCD type 0 together with other SCD types. |

Related scenario

For related scenarios, see tMysqlSCD on page 2563.
tNetsuiteConnection

Creates a connection to the NetSuite SOAP server so that other NetSuite components in the Job can reuse the connection.

**tNetsuiteConnection Standard properties**

These properties are used to configure tNetsuiteConnection running in the Standard Job framework.

The Standard tNetsuiteConnection component belongs to the Business family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Endpoint</strong></td>
<td>Enter the WebService URL required to connect to the NetSuite server.</td>
</tr>
<tr>
<td><strong>API version</strong></td>
<td>Select the API version corresponding to the version of NetSuite you are using.</td>
</tr>
<tr>
<td><strong>E-mail and Password</strong></td>
<td>Enter your user credentials to access the NetSuite SOAP service. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Role</strong></td>
<td>Enter your NetSuite role ID.</td>
</tr>
<tr>
<td><strong>Account</strong></td>
<td>Enter your NetSuite web services account number (also called account ID).</td>
</tr>
<tr>
<td><strong>Application ID</strong></td>
<td>Enter the NetSuite application ID generated for your account. An application ID is required starting from NetSuite version 2015 and above. You can find your application ID or create a new one on NetSuite's Manage Integration page.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enable customizations</strong></td>
<td>Select this check box to instruct the component to retrieve customizations - custom record types and custom fields. With this option enabled, custom record types will be retrieved and can be selected in the Record Type dialog box, and custom fields will be retrieved and included into the schema. This option is enabled by default. You can disable it if you don’t need custom record types or custom fields.</td>
</tr>
</tbody>
</table>

2712
With this option enabled in the tNetsuiteConnection component, all the other components using this connection will have this option enabled.

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

---

### Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

---

### Usage

| Usage rule | This component is more commonly used with other NetSuite components. |

---

### Related scenario

No scenario is available for this component yet.
tNetsuiteInput

Invokes the NetSuite SOAP service and retrieves data according to the conditions you specify.

**tNetsuiteInput Standard properties**

These properties are used to configure tNetsuiteInput running in the Standard Job framework.

The Standard tNetsuiteInput component belongs to the Business and the Cloud families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select a tNetsuiteConnection component to reuse its connection details to set up the connection to the NetSuite server. With a tNetsuiteConnection selected, the settings related to connection setup will not be displayed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endpoint</td>
<td>Enter the WebService URL required to connect to the NetSuite server.</td>
</tr>
<tr>
<td>API version</td>
<td>Select the API version corresponding to the version of NetSuite you are using.</td>
</tr>
<tr>
<td>E-mail and Password</td>
<td>Enter your user credentials to access the NetSuite SOAP service. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Role</td>
<td>Enter your NetSuite role ID.</td>
</tr>
<tr>
<td>Account</td>
<td>Enter your NetSuite web services account number (also called account ID).</td>
</tr>
<tr>
<td>Application ID</td>
<td>Enter the NetSuite application ID generated for your account. An application ID is required starting from NetSuite version 2015 and above. You can find your application ID or create a new one on NetSuite’s Manage Integration page.</td>
</tr>
<tr>
<td>Record Type</td>
<td>Click the [...] button next to the field to open a dialog box, and then select a record type you want to retrieve or select the Use custom object check box and specify a custom record type in the Object Name field.</td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
</tbody>
</table>
Built-In: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-In** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

**Warning:**
The schema of this component is dynamically filled with columns corresponding to the NetSuite service fields, which vary depending on the **Record Type** selected. You can delete columns that are not relevant to your operation, but do not rename any schema columns or this may cause execution errors.

<table>
<thead>
<tr>
<th>Search conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define search conditions to match fields for data retrieval. Click the [+] button to add as many rows as you need, each row for a condition, and define each of your conditions:</td>
</tr>
<tr>
<td>• <strong>Field</strong>: select a field to search based on the condition.</td>
</tr>
<tr>
<td>• <strong>Operator</strong>: select an operator for the field search condition.</td>
</tr>
<tr>
<td>• <strong>Value</strong>: enter the value for the field search condition.</td>
</tr>
<tr>
<td>• <strong>Value 2</strong>: enter the second value for the field search condition, if relevant depending on the operator you select.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Advanced settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Body fields only</strong></td>
</tr>
<tr>
<td>Select this check box to retrieve only body fields. Clear this check box to retrieve body fields and sublist values.</td>
</tr>
<tr>
<td><strong>Enable customizations</strong></td>
</tr>
<tr>
<td>Select this check box to instruct the component to retrieve customizations - custom record types and</td>
</tr>
</tbody>
</table>
custom fields. With this option enabled, custom record types will be retrieved and can be selected in the **Record Type** dialog box, and custom fields will be retrieved and included into the schema.

This option is enabled by default. You can disable it if you don’t need custom record types or custom fields.

<table>
<thead>
<tr>
<th><strong>tStatCatcher Statistics</strong></th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

**Global Variables**

| **Global Variables** | **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |
|----------------------|------------------------------------------------------------------------------------------------------------------|

**Usage**

<table>
<thead>
<tr>
<th><strong>Usage rule</strong></th>
<th>This component is usually used as a start component in a <em>Talend</em> flow and it requires an output component.</th>
</tr>
</thead>
</table>

**Scenario: Handling data with NetSuite**

This scenario describes a Job that reads data from NetSuite, transforms the data and writes the transformed data to NetSuite, finally displays the transformed data on the console.
Adding the components

Create a new Job and add the following five components by typing their names in the design workspace or dropping them from the Palette:

- two tNetsuiteInput components: to read data that matches the specified search criteria from the NetSuite server.
- a tMap component: to transform the input data to the output structure.
- a tNetsuiteOutput component: to write the transformed data into the NetSuite server.
- a tLogRow component: to display the data received from the NetSuite server.

Configuring and linking the components

Reading, transforming, and writing data in NetSuite

Procedure

1. Double-click the first tNetsuiteInput to open its Basic settings view.
2. In the Endpoint, API version, E-mail, Password, Role, Account, and Application ID fields, enter the information required to access NetSuite.

3. Click the […] button next to the Record Type field, and from the dialog box that opens select Location to read the location type information from NetSuite.

   The schema corresponding to the location record type is automatically retrieved.

4. Define a search condition.
   a) Click the [+button below the Search conditions table to add one row.
   b) Click the cell of the Field column and select internalId from the drop-down list.
   c) Click the cell of the Operator column and select List - Any Of from the drop-down list.
   d) In the cell of the Value column, enter java.util.Arrays.asList("1") to retrieve only the record whose internalId is set to 1.

5. Connect the first tNetsuiteInput to tMap using a Row > Main connection.

6. Double-click tNetsuiteOutput to open its Basic settings view.
7. In the Endpoint, API version, E-mail, Password, Role, Account, and Application ID fields, enter the information required to access NetSuite.

8. In the Action list, select insert to insert the transformed data into NetSuite.

9. Click the [...] button next to the Record Type field, and in the dialog box that opens select Location to write the location type information into NetSuite.

10. Click the [...] button next to Edit schema to open the schema editor. You can see the schema has already been prefilled. Keep only the columns that are relevant to your operation and delete others. In this example, keep the column Name, Parent, IncludeChildren, IsInactive, TranPrefix, Logo, MakeInventoryAvailable, MakeInventoryAvailableStore, CustomFieldList, and ExternalId.

11. Connect tMap to tNetsuiteOutput using a Row > *New Output* (Main) connection. In the pop-up dialog box, enter the name of the output connection, out in this example. Then click Yes to let tMap get the schema of the target component tNetsuiteOutput.

12. Double-click tMap to open its Map Editor and configure data flow mappings.
a) In the **out** output table, set the value of the **Name** column by entering “**Paris**” in the corresponding **Expression** column field.

b) Drag and drop the **Parent** column in the **row1** input table onto the **Expression** column of the corresponding row in the **out** output table.

c) Do the same to drag and drop other columns **IncludeChildren**, **IsInactive**, **TranPrefix**, **Logo**, **MakeInventoryAvailable**, **MakeInventoryAvailableStore**, **CustomFieldList**, and **ExternalId** in the **row1** input table onto the **Expression** column of the corresponding row in the **out** output table.

**Retrieving the transformed data from NetSuite**

**Procedure**

1. Double-click the second **tNetsuitInput** to open its **Basic settings** view.
2. In the Endpoint, API version, E-mail, Password, Role, Account, and Application ID fields, enter the information required to access NetSuite.

3. Click the […] button next to the Record Type field, and from the dialog box that opens select Location to read the location type information from NetSuite.
   The schema corresponding to the location record type is automatically retrieved.

4. Define a search condition.
   a) Click the [+ ] button below the Search condition table to add one row.
   b) Click the cell of the Field column and select name from the drop-down list.
   c) Click the cell of the Operator column and select String - Contains from the drop-down list.
   d) In the cell of the Value column, enter Paris to retrieve only the record whose name contains Paris.

5. Double-click tLogRow to open its Basic settings view.

6. In the Mode area, select Vertical (each row is a key/value list) for better readability of the result.

7. Connect the second tNetsuiteInput to tLogRow using a Row > Main connection.

8. Connect the first tNetsuiteInput to the second tNetsuiteInput using a Trigger > OnSubjobOk connection.
Saving and executing the Job

Procedure

1. Press **Ctrl + S** to save the Job.
2. Press **F6** or click **Run** on the **Run** tab to run the Job.

As shown above, the data has been transformed and written into NetSuite, then the transformed data is retrieved from NetSuite and displayed on the console.
tNetsuiteOutput

Invokes the NetSuite SOAP service and inserts, updates, or removes data on the NetSuite SOAP server.

**tNetsuiteOutput Standard properties**

These properties are used to configure tNetsuiteOutput running in the Standard Job framework.

The Standard tNetsuiteOutput component belongs to the Business and the Cloud families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select a tNetsuiteConnection component to reuse its connection details to set up the connection to the NetSuite server. With a tNetsuiteConnection selected, the settings related to connection setup will not be displayed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endpoint</td>
<td>Enter the WebService URL required to connect to the NetSuite server.</td>
</tr>
<tr>
<td>API version</td>
<td>Select the API version corresponding to the version of NetSuite you are using.</td>
</tr>
<tr>
<td>E-mail and Password</td>
<td>Enter your user credentials to access the NetSuite SOAP service. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Role</td>
<td>Enter your NetSuite role ID.</td>
</tr>
<tr>
<td>Account</td>
<td>Enter your NetSuite web services account number (also called account ID).</td>
</tr>
<tr>
<td>Application ID</td>
<td>Enter the NetSuite application ID generated for your account. An application ID is required starting from NetSuite version 2015 and above. You can find your application ID or create a new one on NetSuite’s Manage Integration page.</td>
</tr>
<tr>
<td>Record Type</td>
<td>Click the [...] button next to the field to open a dialog box, and then select a record type you want to retrieve or select the Use custom object check box and specify a custom record type in the Object Name field.</td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
</tbody>
</table>
**Built-In:** You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository:** You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

**Warning:**

The schema of this component is prefilled with columns corresponding to the NetSuite service fields, which vary depending on the **Record Type** selected. You can delete columns that are not relevant to your operation, but do not rename any schema columns or this may cause execution errors.

<table>
<thead>
<tr>
<th>Action</th>
<th>Select an action to perform from the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Adds a new record to your data. If a duplicate is found and the <strong>Die on error</strong> check box is selected, the Job stops; if the <strong>Die on error</strong> check box is cleared, the duplicate is ignored and the Job continues.</td>
</tr>
<tr>
<td><strong>Update</strong></td>
<td>Makes changes to an existing record. If the record corresponding to the specified <strong>Internal ID</strong> does not exist, and the <strong>Die on error</strong> check box is selected, the Job stops. To use this option, the <strong>InternalId</strong> column must exist in the schema and set as Key.</td>
</tr>
</tbody>
</table>
### tNetsuiteOutput

- **Upsert**: Updates the record corresponding to the specified ID, or inserts a new record if it does not exist.
  
  To use this option, a column for record ID must exist in the schema and set as Key.

- **Delete**: Removes an existing record.
  
  If the record corresponding to the specified Internal ID does not exist and the **Die on error** check box is selected, the Job stops.

  To use this option, the **InternalId** column must exist in the schema and set as Key.

### Sync outgoing schema

After the component schema is updated, click this button propagate the schema to the outgoing flow(s), which can be a normal flow, a reject flow, or both, if the component has any.

### Advanced settings

#### Enable customizations

Select this check box to instruct the component to retrieve customizations - custom record types and custom fields. With this option enabled, custom record types will be retrieved and can be selected in the **Record Type** dialog box, and custom fields will be retrieved and included into the schema.

This option is enabled by default. You can disable it if you don’t need custom record types or custom fields.

#### Use native Upsert operation

Select this check box to use the Upsert operation provided by NetSuite.

To use NetSuite’s Upsert operation, the **ExternalId** column must exist in the schema and set as Key.

This option is available when **Upsert** is selected from the **Action** list in **Basic settings**.

By default, for compatibility reasons, this option is disabled and the component performs the Add or Update action for an upsert depending on whether the record ID is specified.

#### Batch size

Specify the number of records to be processed in each batch.

Note that batching will be disabled if the output component has an outgoing flow.

#### tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.
### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CURRENT_INTERNALID: The internal ID of the record currently inserted or processed. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

| Usage rule | This component is usually used as an end component in a Talend flow and it requires an input component. |

### Related scenario

For a related scenario, see Scenario: Handling data with NetSuite on page 2716.
tNormalize

Normalizes the input flow following SQL standard to help improve data quality and thus eases the data update.

tNormalize Standard properties

These properties are used to configure tNormalize running in the Standard Job framework.

The Standard tNormalize component belongs to the Processing family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word “line” when naming the fields. Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [<strong>Repository Content</strong>] window.</td>
<td></td>
</tr>
<tr>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
<td></td>
</tr>
<tr>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
<td></td>
</tr>
<tr>
<td>Column to normalize</td>
<td>Select the column from the input flow which the normalization is based on.</td>
</tr>
<tr>
<td>Item separator</td>
<td>Enter the separator which will delimit data in the input flow.</td>
</tr>
</tbody>
</table>

**Note:**
The item separator is based on regular expressions, so the character `.` (a special
Normalization character for regular expression) should be avoided or used carefully here.

### Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get rid of duplicated rows from output</td>
<td>Select this check box to deduplicate rows in the data of the output flow.</td>
</tr>
<tr>
<td>Use CSV parameters</td>
<td>Select this check box to include CSV specific parameters such as escape mode and enclosure character.</td>
</tr>
<tr>
<td>Discard the trailing empty strings</td>
<td>Select this check box to discard the trailing empty strings.</td>
</tr>
<tr>
<td>Trim resulting values</td>
<td>Select this check box to trim leading and trailing whitespace from the resulting data.</td>
</tr>
</tbody>
</table>

**Note:**
When both Discard the trailing empty string and Trim resulting values check boxes are selected, the former works first.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td>NB_LINE</td>
<td>the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage rule</td>
<td>This component can be used as intermediate step in a data flow.</td>
</tr>
<tr>
<td>Limitation</td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the</td>
</tr>
</tbody>
</table>
**Scenario: Normalizing data**

This simple scenario illustrates a Job that normalizes a list of tags for Web forum topics, and displays the result in a table on the Run console.

This list is not well organized and it contains trailing empty strings, leading and trailing whitespace, and repeated tags, as shown below.

```
ldap,
  db2, jdbc driver,
grid computing, talend architecture ,
content, environment,,
tmap,,
eclipse,
database, java, postgresql,
tmap,
database, java, sybase,
deployment,,
repository,
database, informix, java
```

**Setting up the Job**

**Procedure**

1. Drop the following components from the Palette to the design workspace: tFileInputDelimited, tNormalize, tLogRow.
2. Connect the components using Row > Main connections.

![Diagram of component connections]

**Configuring the components**

**Procedure**

1. Double-click the tFileInputDelimited component to open its Basic settings view.
2. In the **File name** field, specify the path to the input file to be normalized.

3. Click the [...] button next to **Edit schema** to open the [Schema] dialog box, and set up the input schema by adding one column named **Tags**. When done, click **OK** to validate your schema setup and close the dialog box, leaving the rest of the settings as they are.

4. Double-click the **tNormalize** component to open **Basic settings** view.

5. Check the schema, and if necessary, click **Sync columns** to get the schema synchronized with the input component.

6. Define the column the normalization operation is based on.

   In this use case, the input schema has only one column, **Tags**, so just accept the default setting.
7. In the **Advanced settings** view, select the **Get rid of duplicate rows from output**, **Discard the trailing empty strings**, and **Trim resulting values** check boxes.

8. In the **tLogRow** component, select the **Print values in the cells of table** radio button.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Click **Run** on the Run tab or press **F6** to execute the Job.

The list is tidied up, with duplicate tags, leading and trailing whitespace and trailing empty strings removed, and the result is displayed in a table cell on the console.
tOleDbInput

Executes a DB query with a strictly defined order which must correspond to the schema definition and passes on the field list to the next component via a Main row link.

This component reads a database and extracts fields based on a query. It offers the flexibility benefit of the DB query and covers all possible SQL queries.

**tOleDbInput Standard properties**

These properties are used to configure tOleDbInput running in the Standard Job framework.

The Standard tOleDbInput component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <em>line</em> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
<td></td>
</tr>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td>Table Name</td>
<td>Enter the name of the table to be processed.</td>
</tr>
<tr>
<td><strong>Query Type</strong></td>
<td>Select either <strong>Built-in</strong> or <strong>Repository</strong> query type from the drop-down list.</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>Fill in manually the query statement or build it graphically using SQLBuilder.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the relevant query stored in the Repository. The <strong>Query</strong> field gets accordingly filled in.</td>
</tr>
</tbody>
</table>

**Guess Query**

Click the button to generate the query which corresponds to the defined table schema in the **Query** field.

**Guess schema**

Click the button to retrieve the schema from the table.

**Query**

Specify the database query paying particularly attention to properly sequence the fields in order to match the schema definition.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Trim all the String/Char columns</strong></th>
<th>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trim column</strong></td>
<td>Select the check box in <strong>Trim column</strong> to remove leading and trailing whitespace from the corresponding column. This field appears only when the <strong>Trim all the String/Char columns</strong> check box is cleared.</td>
</tr>
</tbody>
</table>

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

### Global Variables

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>QUERY</strong>: the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*. |
Usage

| Usage rule | This component is used as a start component in a Job and it always needs an output link. |

Related scenario

For similar scenarios, see tMysqlInput on page 2488.
tOleDbOutput

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

This component writes, updates, makes changes or suppresses entries in a database.

**tOleDbOutput Standard properties**

These properties are used to configure tOleDbOutput running in the Standard Job framework.

The Standard tOleDbOutput component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Enter the connection string used to open the database. For details, please see <a href="http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbconnection.connectionstring.aspx">http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbconnection.connectionstring.aspx</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Specify the name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
<tr>
<td>Action on data</td>
<td>Select one of the following actions that you want to perform on the data from the drop-down list.</td>
</tr>
<tr>
<td></td>
<td>• Insert: insert data.</td>
</tr>
<tr>
<td></td>
<td>• Update: update data.</td>
</tr>
<tr>
<td></td>
<td>• Insert or update: insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td>• Update or insert: update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
</tr>
<tr>
<td></td>
<td>• Delete: delete data.</td>
</tr>
</tbody>
</table>

**Warning:**

It is necessary to specify at least one column as a primary key on which the Update or Delete operation is based. You can do that by clicking **Edit schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the Update or Delete operation. To do that, select the **Use field options** check box and then in the **Key in update** or **Key in delete** column, select the check box(es) next to the column name(s) you want to use as a base for the Update or Delete operation.
<table>
<thead>
<tr>
<th><strong>Clear data in table</strong></th>
<th>Select this check box to delete data in the defined table before any operation.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Additional columns</strong></th>
<th>This option allows you to call SQL functions to perform actions on columns, provided that these are not insert, update or delete actions, or actions that require pre-processing.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Name</strong>: Type in the name of the schema column to be altered or inserted.</td>
</tr>
<tr>
<td></td>
<td>• <strong>SQL expression</strong>: Type in the SQL statement to be executed in order to alter or insert the data in the corresponding column.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Position</strong>: Select <strong>Before</strong>, <strong>After</strong> or <strong>Replace</strong>, depending on the action to be performed on the reference column.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Reference column</strong>: Type in a reference column used to locate or replace the new column, or the column to be modified.</td>
</tr>
<tr>
<td></td>
<td>This option is not available if you have just created the DB table (even if you delete it beforehand).</td>
</tr>
<tr>
<td>Use field options</td>
<td>Select this check box to customize a request, especially if multiple actions are being carried out on the data.</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Field options</td>
<td>Select the check box(es) in the table if you want to use the corresponding schema column(s) as a base for your operation.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Key in update</strong>: define the primary key to use in an update operation.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Key in delete</strong>: define the primary key to use in a delete operation.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Updatable</strong>: define the column(s) to update.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Insertable</strong>: define the column(s) to insert.</td>
</tr>
<tr>
<td></td>
<td>This field appears only when the <strong>Use field options</strong> check box is selected.</td>
</tr>
<tr>
<td>Enable debug mode</td>
<td>Select this check box to display each step involved in the process of writing data in the database.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_UPDATED</strong>: the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_DELETED</strong>: the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_REJECTED</strong>: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>QUERY</strong>: the query statement processed. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.
Usage

| Usage rule | This component is normally an end component of a Job and always needs an input link. |

Related scenario

For similar scenarios, see tMysqlOutput on page 2515.
tOleDbRow

Acts on the actual database structure or on the data depending on the nature of the query and the database.

tOleDbRow is the specific component for this database query. It executes the SQL query stated in the specified database.

**tOleDbRow Standard properties**

These properties are used to configure tOleDbRow running in the Standard Job framework.

The Standard tOleDbRow component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>View schema:</strong> choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td><strong>Change to built-in property:</strong> choose this option to change the schema to <code>Built-in</code> for local changes.</td>
<td></td>
</tr>
<tr>
<td><strong>Update repository connection:</strong> choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td><strong>Table Name</strong></td>
<td>Name of the table to be processed.</td>
</tr>
</tbody>
</table>
### Query type

<table>
<thead>
<tr>
<th>Query type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: Fill in manually the query statement or build it graphically using SQLBuilder</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</td>
<td></td>
</tr>
</tbody>
</table>

### Guess Query

- Click the **Guess Query** button to generate the query which corresponds to your table schema in the **Query** field.

### Query

Enter your database query paying particularly attention to properly sequence the fields in order to match the schema definition.

### Die on error

- This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

### Advanced settings

#### Propagate QUERY's recordset

- Select this check box to insert the result of the query in a COLUMN of the current flow. Select this column from the **use column** list.

  - **Note:**
    - This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY's recordset should be set to the type of **Object** and this component is usually followed by **tParseRecordSet**.

#### Use PreparedStatement

- Select this check box if you want to query the database using a PreparedStatement. In the **Set PreparedStatement Parameter** table, define the parameters.

  - **Parameter Index**: Enter the parameter position in the SQL instruction.
  - **Parameter Type**: Enter the parameter type.
  - **Parameter Value**: Enter the parameter value.

  - **Note:**
    - This option is very useful if you need to execute the same query several times. Performance levels are increased

#### tStatCatcher Statistics

- Select this check box to collect log data at the component level.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>QUERY: the query statement being processed. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
</table>
**tOleDbRow**

**ERROR MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

---

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility of the database query and covers all possible SQL queries.</th>
</tr>
</thead>
</table>

**Dynamic settings**

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections on page 2497* and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489*. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

---

**Related scenario**

For similar scenarios, see *tMysqlRow on page 2548*. 
**tOpenbravoERPInput**

Extracts data from OpenBravoERP database according to the conditions defined in specific columns. tOpenbravoERPInput connects to an OpenbravoERP database entity via the appropriate Web service.

**tOpenbravoERPInput Standard properties**

These properties are used to configure tOpenbravoERPInput running in the Standard Job framework. The Standard tOpenbravoERPInput component belongs to the Business family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Openbravo REST WebService URL</strong></th>
<th>Enter the URL of the Web service that allows you to connect to the OpenbravoERP database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Username and Password</strong></td>
<td>User authentication information. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Entity</strong></td>
<td>Select the appropriate entity from the drop-down list.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the Repository Content window.</td>
</tr>
<tr>
<td></td>
<td>Click Sync columns to retrieve the schema from the previous component connected in the Job.</td>
</tr>
<tr>
<td><strong>WHERE Clause</strong></td>
<td>Enter your WHERE clause.</td>
</tr>
</tbody>
</table>
| Order by | Select this check bow to define how to order the results (the elements in the drop-down list depend on the entity selected)  
|**Sort:** Choose whether to organise the results in either **Ascending** or **Descending** order. |
| First result | Enter the row number you want to retrieve first. |
| Max result | Enter the maximum number of results you want to retrieve. |

**Advanced settings**

| Advanced separator (for numbers) | Select this check box to modify the separators to be used for the numbers. Either:  
|**Thousands separator**  
or  
|**Decimal separator** |
| tStatCatcher Statistics | Select this check box to collect the log data at a component level. |

**Global Variables**

| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
| **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see **Talend Studio User Guide**. |

**Usage**

| Usage rule | This component is generally used as an input component. An output component is required. |
Related Scenario

For a scenario in which `tOpenbravoERPInput` might be used, see Scenario: Writing data in a Microsoft CRM database and putting conditions on columns to extract specified rows on page 2252.
tOpenbravoERPOutput

Writes data in an OpenbravoERP database.

tOpenbravoERPOutput writes an object in an OpenbravoERP database via the appropriate Web service.

**tOpenbravoERPOutput Standard properties**

These properties are used to configure tOpenbravoERPOutput running in the Standard Job framework. The Standard tOpenbravoERPOutput component belongs to the Business family.

The component in this framework is generally available.

**Basic settings**

| **Openbravo REST Webservice URL** | Enter the URL of the Web service that allows you to connect to the OpenbravoERP database. |
| **Username and Password** | User authentication information. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Action on data** | From the list, select the one of the following actions: Update/Create or Remove |
| **Use existing data file** | Select this check box if desired and then select the file by browsing your directory. |
| **Entity** | Select the appropriate entity from the drop-down list. |
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available: • View schema: choose this option to view the schema only. • Change to built-in property: choose this option to change the schema to Built-in for local changes. • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No. |
upon completion and choose this schema metadata again in the [Repository Content] window.
Click Sync columns to retrieve the schema from the previous component connected in the Job.

Advanced settings

**tStatCatcher Statistics**
Select this check box to collect the log data at a component level.

Global Variables

**Global Variables**

**NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

**NB_LINE_UPDATED**: the number of rows updated. This is an After variable and it returns an integer.

**NB_LINE_CREATED**: the number of rows created. This is an After variable and it returns an integer.

**NB_LINE_REMOVED**: the number of rows removed. This is an After variable and it returns an integer.

**NB_LINE_UNAUTHORIZED**: the number of rows unauthorized. This is an After variable and it returns an integer.

**NB_LINE_FAILED**: the number of rows failed. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

**Usage rule**
This component is used as an output component. It requires an input component.

Related scenario

For a scenario in which tOpenbravoERPOutput may be used, see Scenario: Writing data in a Microsoft CRM database and putting conditions on columns to extract specified rows on page 2252.
tOracleBulkExec

Offers gains in performance during operations performed on data of an Oracle database.

tOracleBulkExec inserts, appends, replaces or truncate data in an Oracle database.

The tOracleOutputBulk and tOracleBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tOracleOutputBulkExec component, detailed in a separate section. The advantage of using two separate steps is that the data can be transformed before it is loaded in the database.

**tOracleBulkExec Standard properties**

These properties are used to configure tOracleBulkExec running in the Standard Job framework.

The Standard tOracleBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Connection type</th>
<th>Drop-down list of available drivers:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Oracle OCI</strong></td>
<td>Select this connection type to use Oracle Call Interface with a set of C-language software APIs that provide an interface to the Oracle database.</td>
</tr>
<tr>
<td><strong>Oracle Service Name</strong></td>
<td>Select this connection type to use the TNS alias that you give when you connect to the remote database.</td>
</tr>
<tr>
<td><strong>Oracle SID</strong></td>
<td>Select this connection type to uniquely identify a particular database on a system.</td>
</tr>
<tr>
<td><strong>Oracle Custom</strong></td>
<td>Select this connection type to access a clustered database.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB Version</th>
<th>Select the Oracle version in use.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Host</th>
<th>IP address of the database server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Port number listening the database server.</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Database</th>
<th>Database name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>Schema name.</td>
</tr>
<tr>
<td>----------</td>
<td>---------------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Username and Password</th>
<th>DB user authentication data.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>Name of the table to be written. Note that only one table can be written at a time.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Action on table</th>
<th>On the table defined, you can perform one of the following operations:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>None</strong></td>
<td>No operation is carried out.</td>
</tr>
<tr>
<td><strong>Drop and create table</strong></td>
<td>The table is removed and created again.</td>
</tr>
<tr>
<td><strong>Create table</strong></td>
<td>The table does not exist and gets created.</td>
</tr>
<tr>
<td><strong>Create table if not exists</strong></td>
<td>The table is created if it does not exist.</td>
</tr>
<tr>
<td><strong>Drop table if exists and create</strong></td>
<td>The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td><strong>Clear table</strong></td>
<td>The table content is deleted.</td>
</tr>
<tr>
<td><strong>Truncate table</strong></td>
<td>The table content is deleted. You do not have the possibility to rollback the operation.</td>
</tr>
<tr>
<td>Data file name</td>
<td>Name of the file to be loaded.</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td></td>
<td><strong>Warning:</strong></td>
</tr>
<tr>
<td></td>
<td>This file should be located on the same machine as the database server.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action on data</th>
<th>On the data of the table defined, you can perform:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert:</td>
<td>Insert data to an empty table. If the table already contains data, the Job stops and the existing data remains in the table.</td>
</tr>
<tr>
<td>Update:</td>
<td>Update the existing data. You have to set the key on the schema if this option is selected.</td>
</tr>
<tr>
<td>Append:</td>
<td>Append data to the table, whether the table is empty or not.</td>
</tr>
<tr>
<td>Replace:</td>
<td>if the table already contains data, delete all the existing data and insert the new data. If the table is empty, insert the new data.</td>
</tr>
<tr>
<td>Truncate:</td>
<td>If the table already contains data, truncate all the existing data and insert the new data. If the table is empty, insert the new data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click Edit schema</td>
<td>to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Built-In:</th>
<th>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository:</td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td></td>
<td>When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.</td>
</tr>
</tbody>
</table>
You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

### Advanced settings

<table>
<thead>
<tr>
<th>Advanced separator (for number)</th>
<th>Select this check box to change the separator used for the numbers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use existing control file</td>
<td>Select this check box if you use a control file (.ctl) and specify its path in the .ctl file name field.</td>
</tr>
</tbody>
</table>
| Record format                   | Define the record format:  
|                                 | **Default**: format parameters are set by default.  
|                                 | **Stream**: set Record terminator.  
|                                 | **Fixed**: set the Record length.  
|                                 | **Variable**: set the Field size of the record length. |
| Specify .ctl file’s INTO TABLE clause manually | Select this check box to manually fill in the INTO TABLE clause of the control file. |
| Fields terminated by            | Character, string or regular expression to separate fields:  
|                                 | **None**: no separator is used.  
|                                 | **Whitespace**: the separator used is a space.  
|                                 | **EOF (used for loading LOBs from lobfile)**: the separator used is an EOF character (End Of File).  
|                                 | **Other terminator**: Set another terminator in the Field terminator field. |
| Use fields enclosure            | Select this check box if you want to use enclosing characters for the text:  
|                                 | **Fields enclosure (left part)**: character delimiting the left of the field.  
|                                 | **Field enclosure (right part)**: character delimiting the right of the field. |
| Use schema’s Date Pattern to load Date field | Select this check box to use the date pattern of the schema in the date field. |
| Specify field condition         | Select this check box to define data loading condition. |
| Preserve blanks                 | Select this check box to preserve the blanks. |
| Trailing null columns           | Select this check box to load null columns. |
| Load options                    | Click + to add data loading options:  
|                                 | **Parameter**: select a loading parameter from the list.  
|                                 | **Value**: enter a value for the parameter selected. |
| NLS Language                    | In the list, select the language used for the data that are not used in Unicode. |
Set Parameter NLS_TERRITORY

Select this check box to modify the territory conventions used for day and weeks numbering. Your OS value is the default value used.

Encoding

Select the encoding from the list, or enter the encoding between double quotes if it does not exist in the list. This field is compulsory for database data handling.

Output

Select the type of output for the standard output of the Oracle database:

to console,
to global variable.

Convert columns and table names to uppercase

Select this check box to uppercase the names of the columns and the name of the table.

tStatCatcher Statistics

Select this check box to collect log data at the component level.

Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE_DATA: the number of rows read. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_LINE_BAD: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_INSERTED: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>RETURN_CODE: the return code indicating the result of processing. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

Usage rule

This dedicated component offers performance and flexibility of Oracle DB query handling.

Dynamic settings

Click the [*] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple
connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Limitation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>The database server/client must be installed on the same machine where the Studio is installed or where the Job using <strong>tOracleBulkExec</strong> is deployed, so that the component functions properly.</td>
<td></td>
</tr>
</tbody>
</table>

**Scenario: Truncating and inserting file data into an Oracle database**

This scenario describes how to truncate the content of an Oracle database and load the content of an input file. The related Job is composed of three components that respectively creates the content, output this content into a file to be loaded into the Oracle database after the database table has been truncated.

**Building the Job**

**Procedure**

1. Drop the following components: **tOracleInput**, **tFileOutputDelimited** and **tOracleBulkExec** from the **Palette** to the design workspace.
2. Connect the **tOracleInput** to the **tFileOutputDelimited** using a **Row > Main** link.
3. Connect the **tOracleInput** to the **tOracleBulkExec** using a **OnSubjobOk** trigger link.
Configuring the components

Procedure

1. Double-click the **tOracleInput** to open its **Basic settings** view.
2. Define the Oracle connection details. It is recommended to store the database connection details in the **Metadata** folder of the **Repository** tree view in order to retrieve them easily at any time in any Job.

![tOracleInput 1](image)

3. Define the schema, if it is not stored in the **Repository**. In this example, the schema contains four columns as follows: **ID_Contract, ID_Client, Contract_type** and **Contract_Value**.

4. Define the **tFileOutputDelimited** component parameters, including output **File Name**, **Row separator** and **Fields delimiter**.

5. Double-click on the **tOracleBulkExec** to define the database populating properties.
6. In the Property Type list, select Repository if you stored the database connection details under the Metadata node of the Repository or select Built-in to define them manually. In this scenario, use the Built-in mode.

7. Set the connection parameters in the following fields: Host, Port, Database, Schema, Username, and Password.

8. Fill in the name of the Table to be populated and the Action on data to be carried out. In this use case, select insert.

9. In the Schema list, select Built-in, and click the [...] button next to the Edit schema field to define the structure of the data to be passed to the next component.

10. Click the Advanced settings view to configure the advanced settings of the component.
11. Select the **Use an existing control file** check box if you want to use a control file (.ctl) storing the status of the physical structure of the database. Or, fill in the following fields manually: **Record format**, **Specify .ctl file’s INTO TABLE clause manually**, **Field terminated by**, **Use field enclosure**, **Use schema’s Date Pattern to load Date field**, **Specify field condition**, **Preserve blanks**, **Trailing null columns**, **Load options**, **NLS Language** and **Set Parameter NLS_TERRITORY** according to your database.

12. In the **Encoding** list, select the encoding, or enter the encoding between double quotes if it does not exist in the list.

13. In the **Output** list, select **to console** to output the standard output of the database in the console.

### Executing the Job

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Press **F6** to run the Job. The log is shown in the console of the **Run** view and the table is populated with the parameter file data.

**Results**

Related topic: see [Scenario: Inserting data in bulk in MySQL database](#) on page 2544.
tOracleClose

Closes the transaction committed in the connected Oracle database.

**tOracleClose Standard properties**

These properties are used to configure tOracleClose running in the Standard Job framework.

The Standard tOracleClose component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the connection component of interest if more than one connection is planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is to be used along with Oracle components, especially with tOracleConnection and tOracleCommit. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: |
Related scenarios

No scenario is available for the Standard version of this component yet.
**tOracleCommit**

Validates the data processed through the Job into the connected Oracle database

Using a unique connection, this component commits in one go a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

**tOracleCommit Standard properties**

These properties are used to configure tOracleCommit running in the Standard Job framework.

The Standard tOracleCommit component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tOracleConnection</strong> component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task. <strong>Warning:</strong> If you want to use a <strong>Row &gt; Main</strong> connection to link <strong>tOracleCommit</strong> to your Job, your data will be committed row by row. In this case, do not select the <strong>Close connection</strong> check box or your connection will be closed before the end of your first row commit.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tOracle components, especially with the **tOracleConnection** and **tOracleRollback** components. |
### Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see [Scenario: Reading data from databases through context-based dynamic connections](#) on page 2497 and [Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters](#) on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

## Related scenario

For **tOracleCommit** related scenario, see [Scenario: Inserting data in mother/daughter tables](#) on page 2476
tOracleConnection

Opens a connection to the specified Oracle database for reuse in the subsequent subjob or subjobs.

**tOracleConnection Standard properties**

These properties are used to configure tOracleConnection running in the Standard Job framework. The Standard tOracleConnection component belongs to the Databases and the ELT families. The component in this framework is generally available.

*Note:* This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see *Dynamic database components* on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Connection type</td>
<td>Drop-down list of available drivers:</td>
</tr>
<tr>
<td></td>
<td><strong>Oracle OCI</strong>: Select this connection type to use Oracle Call Interface with a set of C-language software APIs that provide an interface to the Oracle database.</td>
</tr>
<tr>
<td></td>
<td><strong>Oracle Custom</strong>: Select this connection type to access a clustered database.</td>
</tr>
<tr>
<td></td>
<td><strong>Oracle Service Name</strong>: Select this connection type to use the TNS alias that you give when you connect to the remote database.</td>
</tr>
<tr>
<td></td>
<td><strong>WALLET</strong>: Select this connection type to store credentials in an Oracle wallet.</td>
</tr>
<tr>
<td></td>
<td><strong>Oracle SID</strong>: Select this connection type to uniquely identify a particular database on a system.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the Oracle version in use.</td>
</tr>
<tr>
<td>Use tns file</td>
<td>Select this check box to use the metadata of a context included in a tns file.</td>
</tr>
</tbody>
</table>

*Note:* One tns file may have many contexts.
**OracleConnection**

**TNS File**: Enter the path to the tns file manually or browse to the file by clicking the three-dot button next to the field.

**Select a DB Connection in Tns File**: Click the three-dot button to display all the contexts held in the tns file and select the desired one.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Use SSL</strong></td>
<td>Select this check box to use the SSL protocol to protect the communication with your database server.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when <strong>ORACLE CUSTOM</strong> is selected from the <strong>Connection Type</strong> drop-down list and <strong>Oracle 12c</strong> is selected from the <strong>DB Version</strong> drop-down list.</td>
</tr>
<tr>
<td><strong>TrustStore file</strong></td>
<td>The path to the TrustStore file used to authenticate your database server in SSL authentication.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the <strong>Use SSL</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>TrustStore password</strong></td>
<td>The password for the TrustStore file.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the <strong>Use SSL</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Need Client authentication</strong></td>
<td>Select this check box if SSL client authentication is needed.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the <strong>Use SSL</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>KeyStore file</strong></td>
<td>The path to the keystore file used for SSL client authentication.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the <strong>Need Client authentication</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>KeyStore password</strong></td>
<td>The password for the KeyStore file.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>OracleConnection</strong></td>
<td>the password between double quotes and click <strong>OK</strong> to save the settings. This property is available only when the <strong>Need Client authentication</strong> check box is selected.</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Disable CBC protection</strong></td>
<td>Select this check box to disable the use of CBC (Cipher Block Chaining) in SSL authentication. This property is available only when the <strong>Use SSL</strong> check box is selected.</td>
</tr>
</tbody>
</table>
| **Additional JDBC parameters** | Specify additional connection properties for the DB connection you are creating.  

**Note:**  
You can set the encoding parameters through this field. |
| **Use or register a shared DB Connection** | Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the **Shared DB Connection Name** field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.  
This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the **tRunJob** component. Using a shared connection together with a **tRunJob** component with either of these two options enabled will cause your Job to fail.  
This check box is not available when the **Specify a data source alias** check box is selected. |
| **Specify a data source alias** | Select this check box and specify the alias of a data source created on the **Talend Runtime** side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in **Talend Runtime**. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503. |
| **Advanced settings** | **Auto Commit**  
Select this check box to commit any changes to the database automatically upon the transaction.  
With this check box selected, you cannot use the corresponding **commit** component to commit changes to the database; likewise, when using the corresponding **commit** component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding **commit** component.  
Note that the auto commit function commits each SQL statement as a single transaction immediately after the |
statement is executed while the commit component
does not commit only until all of the statements are
executed. For this reason, if you need more room to
manage your transactions in a Job, it is recommended to
use the commit component.

| tStatCatcher Statistics | Select this check box to gather the job processing
metadata at a Job level as well as at each component
level. |

Usage

| Usage rule | This component is more commonly used with other
tOracle\* components, especially with the tOracleCommit
and tOracleRollback components. |

Related scenario

For tOracleConnection related scenario, see tMysqlConnection on page 2475
tOracleInput

Reads an Oracle database and extracts fields based on a query.

tOracleInput executes a database query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Main row link.

**tOracleInput Standard properties**

These properties are used to configure tOracleInput running in the Standard Job framework.

The Standard tOracleInput component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-In or Repository.</td>
</tr>
<tr>
<td>Built-In</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file where the properties are stored.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.</td>
</tr>
<tr>
<td>For more information about setting up and storing database connection parameters, see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Note: When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
<td></td>
</tr>
</tbody>
</table>
2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see *Talend Studio User Guide.*

<table>
<thead>
<tr>
<th>Connection type</th>
<th>Drop-down list of available drivers:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Oracle OCI</strong></td>
<td>Select this connection type to use Oracle Call Interface with a set of C-language software APIs that provide an interface to the Oracle database.</td>
</tr>
<tr>
<td><strong>Oracle Custom</strong></td>
<td>Select this connection type to access a clustered database.</td>
</tr>
<tr>
<td><strong>Oracle Service Name</strong></td>
<td>Select this connection type to use the TNS alias that you give when you connect to the remote database.</td>
</tr>
<tr>
<td><strong>WALLET</strong></td>
<td>Select this connection type to store credentials in an Oracle wallet.</td>
</tr>
<tr>
<td><strong>Oracle SID</strong></td>
<td>Select this connection type to uniquely identify a particular database on a system.</td>
</tr>
</tbody>
</table>

| DB Version | Select the Oracle version in use. |
| Host | Database server IP address. |
| Port | Listening port number of DB server. |
| Database | Name of the database. |
| Oracle schema | Oracle schema name. |

| Username and Password | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields. Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>View schema</strong></td>
<td>choose this option to view the schema only.</td>
</tr>
<tr>
<td><strong>Change to built-in property</strong></td>
<td>choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td><strong>Update repository connection</strong></td>
<td>choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong></td>
</tr>
</tbody>
</table>
upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Table name</th>
<th>Database table name.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query type</strong> and <strong>Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Specify a data source alias</strong></td>
<td>Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503. If you use the component’s own DB configuration, your data source connection will be closed at the end of the component. To prevent this from happening, use a shared DB connection with the data source alias specified. This check box is not available when the Use an existing connection check box is selected.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **Additional JDBC parameters** | Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair, for example, encryption=1;clientname=Talend. This field is not available if the Use an existing connection check box is selected. |
| **tStatCatcher Statistics** | Select this check box to collect log data at the component level. |
| **Use cursor** | Select this check box and in the Cursor size field displayed, specify the number of rows to fetch in one go from the database. The performance can be improved by tuning this fetch size to an appropriate value. |
| **Trim all the String/Char columns** | Select this check box to remove leading and trailing whitespace from all the String/Char columns. |
| **Trim column** | Remove leading and trailing whitespace from defined columns. |
## Global Variables

**NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

---

## Usage

**Usage rule**

This component covers all possible SQL queries for Oracle databases.

**Dynamic settings**

Click the [*] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided.
Scenario 1: Using context parameters when reading a table from an Oracle database

In this scenario, we will read a table from an Oracle database using a context parameter to refer to the table name.

Dropping and linking the components

Procedure
1. Create a new Job and add the following components by typing their names in the design workspace or dropping them from the Palette: a tOracleInput component and a tLogRow component.
2. Connect tOracleInput to tLogRow using a Row > Main link.

Configuring the components

Procedure
1. Double-click tOracleInput to open its Basic Settings view in the Component tab.
2. Select a connection type from the **Connection Type** drop-down list. In this example, it is **Oracle SID**.

Select the version of the Oracle database to be used from the **DB Version** drop-down list. In this example, it is **Oracle 12-7**.

In the **Host** field, enter the Oracle database server’s IP address. In this example, it is **192.168.31.32**.

In the **Database** field, enter the database name. In this example, it is **TALEND**.

In the **Oracle schema** field, enter the Oracle schema name. In this example, it is **TALEND**.

In the **Username** and **Password** fields, enter the authentication details.

3. Click the [...] button next to **Edit schema** to open the schema editor.
4. Click the [+ ] button to add four columns: ID and AGE of the integer type, NAME and SEX of the string type.
   Click OK to close the schema editor and accept the propagation prompted by the pop-up dialog box.

5. Put the cursor in the Table Name field and press F5 for context parameter setting. The dialog box [New Context Parameter] pops up.

   ![New Context Parameter dialog box]

   For more information about context settings, see Talend Studio User Guide.

6. In the Name field, enter the context parameter name. In this example, it is TABLE.
   In the Default value field, enter the name of the Oracle database table to be queried. In this example, it is PERSON.

7. Click Finish to validate the setting.
   The context parameter context.TABLE automatically appears in the Table Name field.

8. In the Query Type list, select Built-In. Then, click Guess Query to get the query statement.

   "SELECT "+context.TABLE+"."ID", "+context.TABLE+"."NAME", "+context.TABLE+"."SEX", "+context.TABLE+"."AGE FROM "+context.TABLE"

9. Double-click tLogRow to open its Basic settings view in the Component tab.
In the **Mode** area, select **Table (print values in cells of a table)** for a better display of the results.

### Saving and executing the Job

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to run the Job.

```plaintext
[statistics] connecting to socket on port 3750
[statistics] connected
<p>| tLogRow_1 |
|------|------|------|</p>
<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>SEX</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ford</td>
<td>Male</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>Rose</td>
<td>Female</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>Sabrina</td>
<td>Female</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>Teddy</td>
<td>Male</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>Kate</td>
<td>Male</td>
<td>35</td>
</tr>
</tbody>
</table>

[statistics] disconnected
```

As shown above, the data in the Oracle database table *PERSON* is displayed on the console.

### Related scenarios

For related scenarios, see:

- **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489.
tOracleOutput

Writes, updates, makes changes or suppresses entries in an Oracle database.
tOracleOutput executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tOracleOutput Standard properties

These properties are used to configure tOracleOutput running in the Standard Job framework.
The Standard tOracleOutput component belongs to the Databases family.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>![icon]</td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.</td>
</tr>
<tr>
<td></td>
<td>For more information about setting up and storing database connection parameters, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

Note:
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component.
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Connection type</th>
<th>Drop-down list of available drivers:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Oracle OCI</strong>: Select this connection type to use Oracle Call Interface with a set of C-language software APIs that provide an interface to the Oracle database.</td>
</tr>
<tr>
<td></td>
<td><strong>Oracle Custom</strong>: Select this connection type to access a clustered database.</td>
</tr>
<tr>
<td></td>
<td><strong>Oracle Service Name</strong>: Select this connection type to use the TNS alias that you give when you connect to the remote database.</td>
</tr>
<tr>
<td></td>
<td><strong>WALLET</strong>: Select this connection type to store credentials in an Oracle wallet.</td>
</tr>
<tr>
<td></td>
<td><strong>Oracle SID</strong>: Select this connection type to uniquely identify a particular database on a system.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB Version</th>
<th>Select the Oracle version in use.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
</tbody>
</table>

**Note:**

The Action on table list will not be available if you select the Enable parallel execution check box in the Advanced settings view.

On the table defined, you can perform one of the following operations:

- **Default**: No operation is carried out.
- **Drop and create table**: The table is removed and created again.
- **Create table**: The table does not exist and gets created.
- **Create table if not exists**: The table is created if it does not exist.
Drop table if exists and create: The table is removed if it already exists and created again.

Clear table: The table content is deleted.

Truncate table: The table content is deleted. You do not have the possibility to rollback the operation.

Truncate table with reuse storage: The table content is deleted. You do not have the possibility to rollback the operation. However, it is allowed to reuse the existing storage allocated to the table though the storage is considered empty.

Warning:
If you select the Use an existing connection check box and select an option other than Default from the Action on table list, a commit statement will be generated automatically before the data insert/update/delete operation.

Action on data
On the data of the table defined, you can perform:

Insert: Add new entries to the table. If duplicates are found, job stops.

Update: Make changes to existing entries

Insert or update: Insert a new record. If the record with the given reference already exists, an update would be made.

Update or insert: Update the record with the given reference. If the record does not exist, a new record would be inserted.

Delete: Remove entries corresponding to the input flow.

Warning:
It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking Edit Schema and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the Advanced settings view where you can simultaneously define primary keys for the Update and Delete operations. To do that: Select the Use field options check box and then in the Key in update column, select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the Key in delete column for the Delete operation.

Schema and Edit schema
A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- View schema: choose this option to view the schema only.
• **Change to built-in property:** choose this option to change the schema to **Built-in** for local changes.

• **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

**Die on error**

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

**Specify a data source alias**

Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

If you use the component’s own DB configuration, your data source connection will be closed at the end of the component. To prevent this from happening, use a shared DB connection with the data source alias specified.

This check box is not available when the **Use an existing connection** check box is selected.

---

### Advanced settings

<table>
<thead>
<tr>
<th>Additional JDBC parameters</th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <strong>Use an existing connection</strong> check box in the Basic settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td></td>
</tr>
</tbody>
</table>
You can press **Ctrl+Space** to access a list of predefined global variables.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Commit every</strong></td>
<td>Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
<tr>
<td><strong>Additional Columns</strong></td>
<td>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Type in the name of the schema column to be altered or inserted as new column.</td>
</tr>
<tr>
<td><strong>SQL expression</strong></td>
<td>Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td>Select Before, Replace or After following the action to be performed on the reference column.</td>
</tr>
<tr>
<td><strong>Reference column</strong></td>
<td>Type in a column of reference that the <strong>tDBOutput</strong> can use to place or replace the new or altered column.</td>
</tr>
<tr>
<td><strong>Use field options</strong></td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td><strong>Use Hint Options</strong></td>
<td>Select this check box to activate the hint configuration area which helps you optimize a query’s execution. In this area, parameters are:</td>
</tr>
<tr>
<td></td>
<td>- <strong>HINT</strong>: specify the hint you need, using the syntax /* + */. <strong>- POSITION</strong>: specify where you put the hint in a SQL statement.</td>
</tr>
<tr>
<td></td>
<td>- <strong>SQL STMT</strong>: select the SQL statement you need to use.</td>
</tr>
<tr>
<td><strong>Convert columns and table to uppercase</strong></td>
<td>Select this check box to set the names of columns and table in upper case.</td>
</tr>
<tr>
<td><strong>Enable debug mode</strong></td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td><strong>Use Batch</strong></td>
<td>Select this check box to activate the batch mode for data processing.</td>
</tr>
<tr>
<td><strong>Batch Size</strong></td>
<td>Specify the number of records to be processed in each batch.. This field appears only when the <strong>Use batch mode</strong> check box is selected.</td>
</tr>
</tbody>
</table>
Support null in “SQL WHERE” statement

Select this check box to validate null in “SQL WHERE” statement.

## Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>( \text{NB_LINE} ): the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \text{NB_LINE_UPDATED} ): the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>( \text{NB_LINE_INSERTED} ): the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>( \text{NB_LINE_DELETED} ): the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>( \text{NB_LINE_REJECTED} ): the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>( \text{QUERY} ): the query statement processed. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

To fill up a field or expression with a variable, press \( \text{Ctrl} + \text{Space} \) to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

## Usage

### Usage rule

This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in an Oracle database. It also allows you to create a reject flow using a \( \text{Row > Rejects} \) link to filter data in error. For an example of \tMysqlOutput in use, see [Scenario: Retrieving data in error with a Reject link on page 2529](#).

### Dynamic settings

Click the [+] button to add a row in the table and fill the \( \text{Code} \) field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The \( \text{Dynamic settings} \) table is available only when the \( \text{Use an existing connection} \) check box is selected in the
### Basic settings

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### Related scenarios

For **tOracleOutput** related topics, see:

- **Scenario: Inserting a column and altering data using tMysqlOutput** on page 2521.
tOracleOutputBulk

Writes a file with columns based on the defined delimiter and the Oracle standards.

The tOracleOutputBulk and tOracleBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tOracleOutputBulkExec component, detailed in a separate section. The advantage of using two separate steps is that the data can be transformed before it is loaded in the database.

tOracleOutputBulk Standard properties

These properties are used to configure tOracleOutputBulk running in the Standard Job framework.

The Standard tOracleOutputBulk component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in:</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository:</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>File Name</td>
<td>Name of the file to be generated.</td>
</tr>
<tr>
<td>Warning:</td>
<td>This file is generated on the local machine or a shared folder on the LAN.</td>
</tr>
<tr>
<td>Append</td>
<td>Select this check box to add the new rows at the end of the file</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository.</td>
</tr>
<tr>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
<td></td>
</tr>
</tbody>
</table>
**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced separator (for number)</td>
<td>Select this check box to change data separators for numbers:</td>
</tr>
<tr>
<td></td>
<td><strong>Thousands separator</strong>: define separators you want to use for thousands.</td>
</tr>
<tr>
<td></td>
<td><strong>Decimal separator</strong>: define separators you want to use for decimals.</td>
</tr>
<tr>
<td>Field separator</td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td>Row separator</td>
<td>String (ex: <code>\n</code> on Unix) to separate rows.</td>
</tr>
<tr>
<td>Encoding</td>
<td>Select the encoding from the list or select Custom and define it manually.</td>
</tr>
<tr>
<td></td>
<td>This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td>Bulk file parameters</td>
<td>Set the parameters <strong>Buffer Size</strong> and <strong>StringBuilder Size</strong> for a performance gain according to the memory size.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the job processing metadata at a job level as well as at each component level.</td>
</tr>
</tbody>
</table>
Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with tOracleBulkExec component. Used together they offer gains in performance while feeding a Oracle database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component family</td>
<td>Databases/Oracle</td>
</tr>
</tbody>
</table>

Related scenarios

For use cases in relation with tOracleOutputBulk, see the following scenarios:

- Scenario: Inserting transformed data in MySQL database on page 2537.
- Scenario: Inserting data in bulk in MySQL database on page 2544.
- Scenario: Truncating and inserting file data into an Oracle database on page 2752.
tOracleOutputBulkExec

Executes the Insert action in the specified Oracle database.

As a dedicated component, it allows gains in performance during Insert operations to an Oracle database.

The tOracleOutputBulk and tOracleBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tOracleOutputBulkExec component.

tOracleOutputBulkExec Standard properties

These properties are used to configure tOracleOutputBulkExec running in the Standard Job framework.

The Standard tOracleOutputBulkExec component belongs to the Databases family.

The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td></td>
<td>Note: When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component</td>
</tr>
</tbody>
</table>
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Connection type</th>
<th><strong>Drop-down list of available drivers:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Oracle OCI:</strong> Select this connection type to use Oracle Call Interface with a set of C-language software APIs that provide an interface to the Oracle database.</td>
</tr>
<tr>
<td></td>
<td><strong>Oracle Service Name:</strong> Select this connection type to use the TNS alias that you give when you connect to the remote database.</td>
</tr>
<tr>
<td></td>
<td><strong>Oracle SID:</strong> Select this connection type to uniquely identify a particular database on a system.</td>
</tr>
<tr>
<td></td>
<td><strong>Oracle Custom:</strong> Select this connection type to access a clustered database.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB Version</th>
<th>Select the Oracle version in use</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address. Currently, only <em>localhost</em>, 127.0.0.1 or the exact IP address of the local machine is allowed for proper functioning. In other words, the database server must be installed on the same machine where the Studio is installed or where the Job using <em>tOracleOutputBulkExec</em> is deployed.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Port</th>
<th>Listening port number of DB server.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Database</th>
<th>Name of the database</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Schema</th>
<th>Name of the schema.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Username and Password</th>
<th>DB user authentication data.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>Name of the table to be written. Note that only one table can be written at a time and that the table must exist for the insert operation to succeed.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Action on table</th>
<th>On the table defined, you can perform one of the following operations:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>None:</strong> No operations is carried out.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create table:</strong> The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Create table:</strong> The table does not exist and gets created.</td>
</tr>
</tbody>
</table>
### Create table if not exists:
The table is created if does not exist.

### Drop table if exists and create:
The table is removed if it already exists and created again.

### Clear table:
The table content is deleted.

### Truncate table:
The table content is deleted. You do not have the possibility to rollback the operation.

<table>
<thead>
<tr>
<th><strong>File Name</strong></th>
<th>Name of the file to be generated and loaded.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Warning:</strong></td>
<td>This file is generated on the machine specified by the URI in the <strong>Host</strong> field so it should be on the same machine as the database server.</td>
</tr>
<tr>
<td><strong>Create directory if not exists</strong></td>
<td>This check box is selected by default. It creates a directory to hold the output table if required.</td>
</tr>
<tr>
<td><strong>Append</strong></td>
<td>Select this check box to add the new rows at the end of the file.</td>
</tr>
</tbody>
</table>

**Action on data**

On the data of the table defined, you can perform:

- **Insert**: Insert data to an empty table. If the table already contains data, the Job stops and the existing data remains in the table.
- **Update**: Update the existing data. You have to set the key on the schema if this option is selected.
- **Append**: Append data to the table, whether the table is empty or not.
- **Replace**: if the table already contains data, delete all the existing data and insert the new data. If the table is empty, insert the new data.
- **Truncate**: If the table already contains data, truncate all the existing data and insert the new data. If the table is empty, insert the new data.

### Schema and Edit Schema

A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No**.
upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

<table>
<thead>
<tr>
<th>Field separator</th>
<th>Character, string or regular expression to separate fields.</th>
</tr>
</thead>
</table>

**Advanced settings**

| Advanced separator (for number) | Select this check box to change data separators for numbers:  
**Thousands separator**: define separators you want to use for thousands.  
**Decimal separator**: define separators you want to use for decimals. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use existing control file</td>
<td>Select this check box and browse to the .ctl control file you want to use.</td>
</tr>
<tr>
<td>Field separator</td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
</tbody>
</table>
| Row separator                 | String (ex: \\
`n` on Unix) to separate rows. |
| Specify .ctl file's INTO TABLE clause manually | Select this check box to enter manually the INTO TABLE clause of the control file directly into the code. |
| Use schema's Date Pattern to load Date field | Select this check box to use the date model indicated in the schema for dates. |
| Specify field condition       | Select this check box to define a condition for loading data. |
| Preserve blanks               | Select this check box to preserve blank spaces. |
| Trailing null columns         | Select this check box to load data with all empty columns. |
| Load options                  | Click + to add data loading options:  
**Parameter**: select a loading parameter from the list.  
**Value**: enter a value for the parameter selected. |
<table>
<thead>
<tr>
<th><strong>NLS Language</strong></th>
<th>From the drop-down list, select the language for your data if the data is not in Unicode.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set Parameter NLS_TERRITORY</strong></td>
<td>Select this check box to modify the conventions used for date and time formats. The default value is that of the operating system.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td><strong>Oracle encoding type</strong></td>
<td>Select the Oracle-specific encoding type for the data to be processed. This encoding type allows <strong>tOracleOutputBulkExec</strong> to properly generate the Sybase Bulk command.</td>
</tr>
</tbody>
</table>
| **Output** | Select the type of output for the standard output of the Oracle database:  
  - to console,  
  - to global variable. |
| **Convert columns and table names to uppercase** | Select this check box to put columns and table names in upper case. |
| **Bulk file parameters** | Set the parameters **Buffer Size** and **StringBuilder Size** for a performance gain according to the memory size. |
| **tStatCatcher Statistics** | Select this check box to gather the job processing metadata at a job level as well as at each component level. |

### Usage

<table>
<thead>
<tr>
<th><strong>Usage rule</strong></th>
<th>This component is mainly used when no particular transformation is required on the data to be loaded onto the database.</th>
</tr>
</thead>
</table>
| **Dynamic settings** | Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.  

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: |
Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

| Limitation | The database server/client must be installed on the same machine where the Studio is installed or where the Job using tOracleOutputBulkExec is deployed, so that the component functions properly. |

Related scenarios

For use cases in relation with tOracleOutputBulkExec, see the following scenarios:

- Scenario: Inserting transformed data in MySQL database on page 2537.
- Scenario: Inserting data in bulk in MySQL database on page 2544.
- Scenario: Truncating and inserting file data into an Oracle database on page 2752.
tOracleRollback

Cancels the transaction commit in the connected Oracle database to avoid committing part of a transaction involuntarily.

**tOracleRollback Standard properties**

These properties are used to configure tOracleRollback running in the Standard Job framework. The Standard tOracleRollback component belongs to the Databases family. The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Component list</strong></td>
<td>Select the tOracleConnection component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td><strong>Close Connection</strong></td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

### Advanced settings

| **tStatCatcher Statistics** | Select this check box to collect log data at the component level. |

### Usage

| **Usage rule** | This component is more commonly used with other tOracle* components, especially with the tOracleConnection and tOracleCommit components. |
| **Dynamic settings** | Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |
Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

## Related scenario

For **tOracleRollback** related scenario, see **tMysqlRollback** on page 2546
tOracleRow

Executes the stated SQL query on the specified Oracle database.

Depending on the nature of the query and the database, tOracleRow acts on the actual DB structure or on the data (although without handling data). The SQLBuilder tool helps you write easily your SQL statements.

tOracleRow is the specific component for this database query. It executes the SQL query stated onto the specified database. The row suffix means the component implements a flow in the job design although it doesn’t provide output.

**tOracleRow Standard properties**

These properties are used to configure tOracleRow running in the Standard Job framework.

The Standard tOracleRow component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
<td></td>
</tr>
</tbody>
</table>
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide.*

| Connection type | Drop-down list of available drivers:  
|----------------|---------------------------------|
| Oracle OCI     | Select this connection type to use Oracle Call Interface with a set of C-language software APIs that provide an interface to the Oracle database.  
| Oracle Service Name | Select this connection type to use the TNS alias that you give when you connect to the remote database.  
| Oracle SID      | Select this connection type to uniquely identify a particular database on a system.  
| Oracle Custom   | Select this connection type to access a clustered database.  

| DB Version | Select the Oracle version in use.  

| Host | Database server IP address.  

| Port | Listening port number of DB server.  

| Database | Name of the database.  

| Username and Password | DB user authentication data.  

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.  

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  

- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.  


<table>
<thead>
<tr>
<th><strong>Built-In:</strong> You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Repository:</strong> You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Query type</strong></th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in:</strong> Fill in manually the query statement or build it graphically using SQLBuilder</td>
<td></td>
</tr>
<tr>
<td><strong>Repository:</strong> Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Query</strong></th>
<th>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Use NB_LINE_</strong></th>
<th>This option allows you feed the variable with the number of rows inserted/updated/deleted to the next component or subjob. This field only applies if the query entered in Query field is a INSERT, UPDATE or DELETE query.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>NONE:</strong> does not feed the variable.</td>
<td></td>
</tr>
<tr>
<td>• <strong>INSERTED:</strong> feeds the variable with the number of rows inserted.</td>
<td></td>
</tr>
<tr>
<td>• <strong>UPDATED:</strong> feeds the variable with the number of rows updated.</td>
<td></td>
</tr>
<tr>
<td>• <strong>DELETED:</strong> feeds the variable with the number of rows deleted.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Specify a data source alias</strong></th>
<th>Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you use the component’s own DB configuration, your data source connection will be closed at the end of the component. To prevent this from happening, use a shared DB connection with the data source alias specified.</td>
<td></td>
</tr>
<tr>
<td>This check box is not available when the Use an existing connection check box is selected.</td>
<td></td>
</tr>
</tbody>
</table>

| **Die on error** | This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row > Rejects link. |
### Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propagate QUERY’s recordset</td>
<td>Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the use column list. Note: This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of Object and this component is usually followed by tParseRecordSet.</td>
</tr>
<tr>
<td>Use PreparedStatement</td>
<td>Select this check box if you want to query the database using a PreparedStatement. In the Set PreparedStatement Parameter table, define the parameters represented by “?” in the SQL instruction of the Query field in the Basic Settings tab. Parameter Index: Enter the parameter position in the SQL instruction. Parameter Type: Enter the parameter type. Parameter Value: Enter the parameter value. Note: This option is very useful if you need to execute the same query several times. Performance levels are increased. You can also use PreparedStatement to avoid SQL injection. For a detailed scenario of utilizing this feature, see Scenario: Using PreparedStatement objects to query data on page 2553.</td>
</tr>
<tr>
<td>Commit every</td>
<td>Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUERY</td>
<td>the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>NB_LINE_UPDATED</td>
<td>the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_INSERTED</td>
<td>the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_DELETED</td>
<td>the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions...</td>
</tr>
</tbody>
</table>
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
</table>

**Dynamic settings**

Click the [*] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Related scenarios

For related topics, see:

- **Scenario: Combining two flows for selective output** on page 2558
- **Procedure** on page 625.
- **Scenario: Removing and regenerating a MySQL table index** on page 2552.
- **Scenario: Using PreparedStatement objects to query data** on page 2553.
tOracleSCD

Reflects and tracks changes in a dedicated Oracle SCD table.
tOracleSCD addresses Slowly Changing Dimension needs, reading regularly a source of data and logging the changes into a dedicated SCD table.

**tOracleSCD Standard properties**

These properties are used to configure tOracleSCD running in the Standard Job framework.
The Standard tOracleSCD component belongs to the Business Intelligence and the Databases families.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the Repository file where properties are stored. The following fields are pre-filled in using fetched data.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
</tbody>
</table>
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Connection type</strong></th>
<th>Drop-down list of available drivers:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Oracle OCI</strong></td>
<td>Select this connection type to use Oracle Call Interface with a set of C-language software APIs that provide an interface to the Oracle database.</td>
</tr>
<tr>
<td><strong>Oracle Service Name</strong></td>
<td>Select this connection type to use the TNS alias that you give when you connect to the remote database.</td>
</tr>
<tr>
<td><strong>Oracle SID</strong></td>
<td>Select this connection type to uniquely identify a particular database on a system.</td>
</tr>
<tr>
<td><strong>Oracle Custom</strong></td>
<td>Select this connection type to access a clustered database.</td>
</tr>
</tbody>
</table>

| **DB Version** | Select the Oracle version you are using. |
| **Host**       | Database server IP address. |
| **Port**       | Listening port number of DB server. |
| **Database**   | Name of the database. |
| **Schema**     | Name of the DB schema. |

| **Username and Password** | DB user authentication data. |
|                          | To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |

| **Table** | Name of the table to be written. Note that only one table can be written at a time. |

| **Action on table** | Select to perform one of the following operations on the table defined: |
|                    | - None: No action is carried out on the table. |
|                    | - Create table: A new table is created. |
|                    | - Create table if not exists: A table is created if it does not exist. |

| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. |
|                           | Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available: |
|                           |   • View schema: choose this option to view the schema only. |
|                           |   • Change to built-in property: choose this option to change the schema to Built-in for local changes. |
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in</td>
<td>The schema is created and stored locally for this component only. Related topic: see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
<tr>
<td>Repository</td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

**SCD Editor**

The SCD editor helps to build and configure the data flow for slowly changing dimension outputs. For more information, see [SCD management methodology](#) on page 2566.

**Use memory saving Mode**

Select this check box to maximize system performance.

**Source keys include Null**

Select this check box to allow the source key columns to have Null values.

**Warning:**

Special attention should be paid to the uniqueness of the source key(s) values when this option is selected.

**Die on error**

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

### Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional JDBC parameters</td>
<td>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the Use an existing connection check box in the Basic settings.</td>
</tr>
<tr>
<td>End date time details</td>
<td>Specify the time value of the SCD end date time setting in the format of HH:mm:ss. The default value for this field is 12:00:00. This field appears only when SCD Type 2 is used and Fixed year value is selected for creating the SCD end date.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
<tr>
<td>Debug mode</td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
</tbody>
</table>
## Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE_UPDATED: the number of rows updated. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_REJECTED</strong>: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as Output component. It requires an Input component and Row main link as input.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</td>
</tr>
<tr>
<td></td>
<td>The <strong>Dynamic settings</strong> table is available only when the Use an existing connection check box is selected in the <strong>Basic settings</strong> view. Once a dynamic parameter is defined, the Component List box in the <strong>Basic settings</strong> view becomes unusable.</td>
</tr>
<tr>
<td></td>
<td>For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on <strong>Dynamic settings</strong> and context variables, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Limitation</td>
<td>This component does not support using SCD type 0 together with other SCD types.</td>
</tr>
</tbody>
</table>
Related scenario

For related scenarios, see tMysqlSCD on page 2563.
tOracleSCDELT

Reflects and tracks changes in a dedicated Oracle SCD table through SQL queries.

*tOracleSCDELT* addresses Slowly Changing Dimension needs through SQL queries (server-side processing mode), and logs the changes into a dedicated Oracle SCD table.

**tOracleSCDELT Standard properties**

These properties are used to configure *tOracleSCDELT* running in the Standard Job framework.

The Standard *tOracleSCDELT* component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

*Note:*

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally. Enter properties manually.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file where Properties are stored. The fields that come after are pre-filled in using the fetched data.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

*Note:*

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide.*

| **Connection type** | Drop-down list of available drivers:  
|---------------------|---|  
| **Oracle OCI**      | Select this connection type to use Oracle Call Interface with a set of C-language software APIs that provide an interface to the Oracle database.  
| **Oracle Service Name** | Select this connection type to use the TNS alias that you give when you connect to the remote database.  
| **Oracle SID**       | Select this connection type to uniquely identify a particular database on a system.  
| **Oracle Custom**    | Select this connection type to access a clustered database.  

| **DB Version** | Select the Oracle version you are using.  
| **Host** | The IP address of the database server.  
| **Port** | Listening port number of database server.  
| **Database** | Name of the database  
| **Username and Password** | User authentication data for a dedicated database.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.  

| **Source table** | Name of the input DB2 SCD table.  
| **Table** | Name of the table to be written. Note that only one table can be written at a time  
| **Action on table** | Select to perform one of the following operations on the table defined:  
| **None** | No action carried out on the table.  
| **Drop and create table** | The table is removed and created again  
| **Create table** | A new table gets created.  
| **Create table if not exists** | A table gets created if it does not exist.  
| **Clear table** | The table content is deleted. You have the possibility to rollback the operation.  
| **Truncate table** | The table content is deleted. You do not have the possibility to rollback the operation.  

| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

| **Built-in** | The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide. |
| **Repository** | The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide. |

### Surrogate Key

Select the surrogate key column from the list.

### Creation

Select the method to be used for the surrogate key generation.

For more information regarding the creation methods, see SCD management methodology on page 2566.

### Source Keys

Select one or more columns to be used as keys, to ensure the unicity of incoming data.

### Source fields value include Null

Select this check box to allow the source columns to have Null values.

**Note:**

The source columns here refer to the fields defined in the **SCD type 1 fields** and **SCD type 2 fields** tables.

### Use SCD Type 1 fields

Use type 1 if tracking changes is not necessary. SCD Type 1 should be used for typos corrections for example. Select the columns of the schema that will be checked for changes.

### Use SCD Type 2 fields

Use type 2 if changes need to be tracked down. SCD Type 2 should be used to trace updates for example. Select the columns of the schema that will be checked for changes.

**Start date:** Adds a column to your SCD schema to hold the start date value. You can select one of the input schema columns as Start Date in the SCD table.

**End Date:** Adds a column to your SCD schema to hold the end date value for the record. When the record is currently active, the End Date column shows a null value,
or you can select Fixed Year value and fill it in with a fictive year to avoid having a null value in the End Date field.

**Log Active Status**: Adds a column to your SCD schema to hold the true or false status value. This column helps to easily spot the active record.

**Log versions**: Adds a column to your SCD schema to hold the version number of the record.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Additional JDBC parameters</strong></th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the <strong>Use an existing connection</strong> check box in the Basic settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Debug mode</strong></td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it. For further information about variables, see <strong>Talend Studio User Guide</strong>.</th>
</tr>
</thead>
</table>

### Usage

<table>
<thead>
<tr>
<th><strong>Usage rule</strong></th>
<th>This component is used as an output component. It requires an input component and Row main link as input.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamic settings</strong></td>
<td>Click the [*] button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</td>
</tr>
</tbody>
</table>
The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

---

**Related Scenario**

For related topics, see **tOracleSCD** on page 2795 and **tMysqlSCD** on page 2563.
**tOracleSP**

Calls an Oracle database stored procedure.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**tOracleSP Standard properties**

These properties are used to configure tOracleSP running in the Standard Job framework.

The component in this framework is generally available.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](#).

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>
| **Connection type** | Drop-down list of available drivers:  
**Oracle OCI**: Select this connection type to use Oracle Call Interface with a set of C-language software APIs that provide an interface to the Oracle database. |
<table>
<thead>
<tr>
<th><strong>Oracle Service Name:</strong></th>
<th>Select this connection type to use the TNS alias that you give when you connect to the remote database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Oracle SID:</strong></td>
<td>Select this connection type to uniquely identify a particular database on a system.</td>
</tr>
<tr>
<td><strong>Oracle Custom:</strong></td>
<td>Select this connection type to access a clustered database.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Property type</strong></th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in:</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository:</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>DB Version</strong></th>
<th>Select the Oracle version in use.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Port</strong></th>
<th>Listening port number of DB server.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Name of the database.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Schema</strong></th>
<th>Name of the schema.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Username and Password</strong></th>
<th>DB user authentication data.</th>
</tr>
</thead>
</table>

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

<table>
<thead>
<tr>
<th><strong>Schema and Edit Schema</strong></th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either <strong>Built-in</strong> or stored remotely in the <strong>Repository</strong>.</th>
</tr>
</thead>
</table>

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No upon completion** and choose this schema metadata again in the **[Repository Content]** window.

<table>
<thead>
<tr>
<th><strong>Built-In</strong></th>
<th>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</th>
</tr>
</thead>
</table>
**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>SP Name</th>
<th>Type in the exact name of the Stored Procedure (or Function)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is Function / Return result in</strong></td>
<td>Select this check box, if the stored procedure is a function and one value only is to be returned. Select on the list the schema column, the value to be returned is based on.</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td>Click the Plus button and select the various <strong>Schema Columns</strong> that will be required by the procedures. Note that the SP schema can hold more columns than there are parameters used in the procedure.</td>
</tr>
</tbody>
</table>

Select the **Type** of parameter:

- **IN**: Input parameter.
- **OUT**: Output parameter/return value.
- **IN OUT**: Input parameter is to be returned as value, likely after modification through the procedure (function).
- **RECORDSET**: Input parameters is to be returned as a set of values, rather than single value.

**Note:**

Check **Scenario: Inserting data in mother/daughter tables** on page 2476 if you want to analyze a set of records from a database table or DB query and return single records.

The Custom Type is used when a Schema Column you want to use is user-defined. Two Custom Type columns are available in the Parameters table. In the first Custom Type column:

- Select the check box in the Custom Type column when the corresponding Schema Column you want to use is of user-defined type.
- If all listed Schema Columns in the Parameters table are of custom type, you can select the check box before Custom Type once for them all.

Select a database type from the DB Type list to map the source database type to the target database type:

- **Auto-Mapping**: Map the source database type to the target database type automatically (default)
- **CLOB**: Character large object.
- **BLOB**: Binary large object.
- **DECIMAL**: Decimal numeric object.
- **NUMERIC**: Character 0 to 9.
- **XMLTYPE**: XML schema type.
Warning:
When mapping a column selected from the Return result in list box or set to the IN or OUT type of parameter to XMLTYPE, make sure that the column is of type String in the schema.

In the second Custom Type column, you can precise what the custom type is. The type may be:
- STRUCT: used for one element.
- ARRAY: used for a collection of elements.

In the Custom name column, specify the name of the custom type that you have given to this type.

Warning:
When an OUT parameter uses the custom type, make sure that its corresponding Schema Column has chosen the Object type in the schema table.

Specify a data source alias
Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

Advanced settings

**Additional JDBC parameters**
Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the Use an existing connection check box in the Basic settings.

**NLS Language**
In the list, select the language used for the data that are not used in Unicode.

**NLS Territory**
Select the conventions used for date and time formats. The default value is that of the operating system.

**tStatCatcher Statistics**
Select this check box to gather the job processing metadata at a Job level as well as at each component level.

Usage

**Usage rule**
This component is used as intermediary component. It can be used as start component but only input parameters are thus allowed.

**Dynamic settings**
Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple
connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

| Limitation | The Stored Procedures syntax should match the Database syntax. When the parameters set in this component are of Custom Type, the tJava family components should be placed before the component in order for users to define values for the custom-type parameters, or after the component so as to read and output the Out-type custom parameters. |

**Scenario: Checking number format using a stored procedure**

The following job aims at connecting to an Oracle Database containing Social Security Numbers and their holders' name, calling a stored procedure that checks the SSN format of against a standard ###-###-#### format. Then the verification output results, 1 for valid format and 0 for wrong format get displayed onto the execution console.

- Drag and drop the following components from the Palette: tOracleConnection, tOracleInput, tOracleSP and tLogRow.
- Link the tOracleConnection to the tOracleInput using a Then Run connection as no data is handled here.
• And connect the other components using a **Row Main** link as rows are to be passed on as parameter to the SP component and to the console.

• In the **tOracleConnection**, define the details of connection to the relevant Database. You will then be able to reuse this information in all other DB-related components.

• Then select the **tOracleInput** and define its properties.

  ![tOracleInput_1](image)

  - Select the **Use an existing connection** check box and select the **tOracleConnection** component in the list in order to reuse the connection details that you already set.

  - Select **Repository** as **Property type** as the Oracle schema is defined in the DB Oracle connection entry of the Repository. If you haven’t recorded the Oracle DB details in the **Repository**, then fill in the Schema name manually.

  - Then select **Repository** as **Schema**, and retrieve the relevant schema corresponding to your Oracle DB table.

  ![Table](image)

  - In this example, the SSN table has a four-column schema that includes **ID**, **NAME**, **CITY** and **SSNUMBER**.

  - In the **Query** field, type in the following **Select query** or select it in the list, if you stored it in the Repository.

    ```sql
    select ID, NAME, CITY, SSNUMBER from SSN
    ```

  - Then select the **tOracleSP** and define its **Basic settings**.
Like for the **tOracleInput** component, select **Repository** in the **Property type** field and select the **Use an existing connection** check box, then select the relevant entries in the respective list.

The schema used for the **tOracleSP** slightly differs from the input schema. Indeed, an extra column (**SSN_Valid**) is added to the Input schema. This column will hold the format validity status (1 or 0) produced by the procedure.

In the **SP Name** field, type in the exact name of the stored procedure (or function) as called in the Database. In this use case, the stored procedure name is **is_ssn**.

The basic function used in this particular example is as follows:

```sql
CREATE OR REPLACE FUNCTION is_ssn(string_in VARCHAR2)
RETURN PLS_INTEGER
IS
  -- validating ###-##-#### format
BEGIN
  IF TRANSLATE(string_in, '0123456789A', 'AAAAAAAAAAB') = 'AAA-AA-AAAA' THEN
    RETURN 1;
  END IF;
  RETURN 0;
END is_ssn;
/
```

As a return value is expected in this use case, the procedure acts as a function, so select the **Is function** check box.

The only return value expected is based on the **ssn_valid** column, hence select the relevant list entry.

In the **Parameters** area, define the input and output parameters used in the procedure. In this use case, only the **SSNNumber** column from the schema is used in the procedure.
Click the plus sign to add a line to the table and select the relevant column (SSNumber) and type (IN).

Then select the tLogRow component and click Sync Column to make sure the schema is passed on from the preceding tOracleSP component.

Select the Print values in cells of a table check box to facilitate the output reading.

Then save your job and press F6 to run it.

On the console, you can read the output results. All input schema columns are displayed even though they are not used as parameters in the stored procedure.

The final column shows the expected return value, whether the SS Number checked is valid or not.

Note:
Check Scenario: Inserting data in mother/daughter tables on page 2476 if you want to analyze a set of records from a database table or DB query and return single records.

Related scenarios

For related scenarios, see:

- Scenario: Retrieving personal information using a stored procedure on page 2452.
- Scenario: Using tMysqlSP to find a State Label using a stored procedure on page 2583.
- Executing a stored procedure using tMDMSP on page 2190.
tOracleTableList

Lists the names of specified Oracle tables using a SELECT statement based on a WHERE clause. tOracleTableList iterates on a set of tables through a defined Oracle connection.

**tOracleTableList Standard properties**

These properties are used to configure tOracleTableList running in the Standard Job framework. The Standard tOracleTableList component belongs to the Databases family. The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tOracleConnection component in the list if more than one connection is planned for the current Job.</td>
</tr>
<tr>
<td>Where clause for table name selection</td>
<td>Enter the WHERE clause that will be used to identify the tables to iterate on.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>CURRENT_TABLE: the name of the table currently iterated upon. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_TABLE: the number of tables iterated upon so far. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with other Oracle components, especially with <strong>tOracleConnection</strong>.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |

Related scenarios

For a **tOracleTableList** related scenario, see *Scenario: Iterating on DB tables and deleting their content using a user-defined SQL template* on page 2588.
tPaloCheckElements

Checks whether elements are present in an incoming data flow existing in a given cube. This component can be used along with tPaloOutputMulti. It checks if the elements from the input stream exist in the given cube, before writing them. It can also define a default value to be used for nonexistent elements.

tPaloCheckElements Standard properties

These properties are used to configure tPaloCheckElements running in the Standard Job framework. The Standard tPaloCheckElements component belongs to the Business Intelligence family. The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Name</td>
<td>Enter the host name or the IP address of the host server.</td>
</tr>
<tr>
<td>Server Port</td>
<td>Type in the listening port number of the Palo server.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Enter the Palo user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Database</td>
<td>Type in the name of the database in which the data is to be written.</td>
</tr>
<tr>
<td>Cube</td>
<td>Type in the name of the cube in which the data should be written.</td>
</tr>
<tr>
<td>On element error</td>
<td>Select what should happen if an element does not exist: - Reject row: the corresponding row is rejected and placed in the reject flow. - Use default: the defined Default value is used. - Stop: the entire process is interrupted.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
</tbody>
</table>
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th><strong>Built-in</strong>: The schema is created and stored locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Repository</strong>: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>

Define the elements to be checked in the table provided.
- **Column**: shows the column(s) from the input schema. It is completed automatically once a schema is retrieved or created.
- **Element type**: select the element type for the input column. Only one column can be defined as **Measure**.
- **Default**: type in the default value to be used if you have selected the **Use default** option in the **On element error** field.

**Advanced settings**

| **tStat Catcher Statistics** | Select this check box to collect log data on the component level. |

**Global Variables**

| **Global Variables** | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. |

---

2816
For further information about variables, see  
*Talend Studio User Guide*.

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component requires an input component.</th>
</tr>
</thead>
</table>

**Connections**

- Outgoing links (from this component to another):
  - **Row**: Main; Rejects
  - **Trigger**: Run if; On Component Ok; On Component Error.

- Incoming links (from one component to this one):
  - **Row**: Main; Rejects

For further information regarding connections, see 
*Talend Studio User Guide*.

**Limitation**

- This component only works on **Normal** Palo cubes.

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

**Related scenario**

For a related scenario, see **Scenario 2: Rejecting inflow data when the elements to be written do not exist in a given cube** on page 2865.
tPaloClose

Closes an active connection to a Palo Server.
This component is used to disconnect a connection to a Palo server to release occupied resources.

tPaloClose Standard properties

These properties are used to configure tPaloClose running in the Standard Job framework.
The Standard tPaloClose component belongs to the Business Intelligence family.
The component in this framework is generally available.

Basic settings

| Component List | Select a tPaloConnection component from the drop-down list if more than one connection is planned for the current Job. |

Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component is more commonly used with other Palo components, especially tPaloConnection. |

| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, |
especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Related scenarios

No scenario is available for the Standard version of this component yet.
tPaloConnection

Opens a connection to a Palo Server and allows other components involved in a process to share the connection for the duration of the process.

**tPaloConnection Standard properties**

These properties are used to configure tPaloConnection running in the Standard Job framework.

The Standard tPaloConnection component belongs to the Business Intelligence family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Name</td>
<td>Enter the host name or the IP address of the host server.</td>
</tr>
<tr>
<td>Server Port</td>
<td>Type in the listening port number of the Palo server.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Enter the Palo user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage rule</td>
<td>This component is used along with Palo components to offer a shared connection to a Palo server.</td>
</tr>
</tbody>
</table>
### Connections

Outgoing links (from this component to another):

**Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.

Incoming links (from one component to this one):

**Row**: Iterate


For further information regarding connections, see *Talend Studio User Guide*.

### Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

---

### Related scenario

For related scenarios, see *Scenario: Creating a dimension with elements* on page 2841.
tPaloCube

Performs operations on a given Palo cube.
This component creates, deletes or clears Palo cubes from existing dimensions in a Palo database.

tPaloCube Standard properties

These properties are used to configure tPaloCube running in the Standard Job framework.
The Standard tPaloCube component belongs to the Business Intelligence family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Name</td>
<td>Enter the host name or the IP address of the host server.</td>
</tr>
<tr>
<td>Server Port</td>
<td>Type in the listening port number of the Palo server.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Enter the Palo user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Database</td>
<td>Type in the name of the database in which the operation is to take place.</td>
</tr>
<tr>
<td>Cube</td>
<td>Type in the name of the cube where the operation is to take place.</td>
</tr>
<tr>
<td>Cube type</td>
<td>From the drop-down list, select the type of cube on which the operation is to be carried out:</td>
</tr>
<tr>
<td></td>
<td>- Normal: this is the normal and default type of cube.</td>
</tr>
<tr>
<td></td>
<td>- Attribute: an Attribute cube is created with a normal cube.</td>
</tr>
<tr>
<td></td>
<td>- User Info: User Info cubes can be created/modified with this component.</td>
</tr>
<tr>
<td>Action on cube</td>
<td>Select the operation you want to carry out on the cube defined:</td>
</tr>
<tr>
<td></td>
<td>- Create cube: the cube does not exist and will be created.</td>
</tr>
<tr>
<td></td>
<td>- Create cube if not exists: the cube is created if it does not exist.</td>
</tr>
</tbody>
</table>
- **Delete cube if exists and create**: the cube is deleted if it already exists and a new one will be created.
- **Delete cube**: the cube is deleted from the database.
- **Clear cube**: the data is cleared from the cube.

### Dimension list
Add rows and enter the name of existing database dimension's to be used in the cube. The order of the dimensions in the list determines the order of the dimensions created.

### Advanced settings

<table>
<thead>
<tr>
<th>tStat Catcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

### Global Variables

| Global Variables | CUBENAME: the name of the cube. This is an After variable and it returns a string.  
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see [Talend Studio User Guide](#). |

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Can be used as a standalone component for dynamic cube creation with a defined dimension list.</th>
</tr>
</thead>
</table>

### Connections

Outgoing links (from this component to another):

**Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.

Incoming links (from one component to this one):

**Row**: Iterate

**Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.

For further information regarding connections, see [Talend Studio User Guide](#).
**Limitation**

The cube creation process does not create dimensions from scratch, so the dimensions to be used in the cube must be created beforehand.

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

---

**Scenario: Creating a cube in an existing database**

The Job in this scenario creates a new two dimensional cube in the Palo demo database *Biker*.

To replicate this scenario, proceed as follows:

**Configuring the tPaloCube component**

**Procedure**

1. Drop **tPaloCube** from the **Palette** onto the design workspace.
2. Double-click **tPaloCube** to open its **Component** view.
3. In the **Host name** field, type in the host name or the IP address of the host server, *localhost* for this example.

4. In the **Server Port** field, type in the listening port number of the Palo server. In this scenario, it is **7777**.

5. In the **Username** field and the **Password** field, type in the authentication information. In this example, both of them are *admin*.

6. In the **Database** field, type in the database name in which you want to create the cube, *Biker* in this example.

7. In the **Cube** field, type in the name you want to use for the cube to be created, for example, *bikerTalend*.

8. In the **Cube type** field, select the **Normal** type from the drop-down list for the cube to be created, meaning this cube will be normal and default.

9. In the **Action on cube** field, select the action to be performed. In this scenario, select **Create cube**.

10. Under the **Dimension list** table, click the plus button twice to add two rows into the table.

11. In the **Dimension list** table, type in the name for each newly added row to replace the default row name. In this scenario, type *Months* for the first row and *Products* for the second. These two dimensions exist already in the Biker database where the new cube will be created.

**Job execution**

Press **F6** to run the Job.

A new cube has been created in the *Biker* database and the two dimensions are added into this cube.
tPaloCubeList

Retrieves a list of cube details from the given Palo database.

This component lists cube names, cube types, number of assigned dimensions, the number of filled cells from the given database.

Discovering the read-only output schema of tPaloCubeList

The below table presents information related to the read-only schema of the tPaloCubeList component.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cube_id</td>
<td>int</td>
<td>Internal id of the cube.</td>
</tr>
<tr>
<td>Cube_name</td>
<td>string</td>
<td>Name of the cube.</td>
</tr>
<tr>
<td>Cube_dimensions</td>
<td>int</td>
<td>Number of dimensions inside the cube.</td>
</tr>
<tr>
<td>Cube_cells</td>
<td>long</td>
<td>Number of calculated cells inside the cube.</td>
</tr>
<tr>
<td>Cube_filled_cells</td>
<td>long</td>
<td>Number of filled cells inside the cube.</td>
</tr>
<tr>
<td>Cube_status</td>
<td>int</td>
<td>Status of the cube. It may be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0: unloaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: loaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: changed</td>
</tr>
<tr>
<td>Cube_type</td>
<td>int</td>
<td>Type of the cube. It may be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0: normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3: user info</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 4: gpu type</td>
</tr>
</tbody>
</table>

The Standard tPaloCubeList component belongs to the Business Intelligence family.

The component in this framework is generally available.

**Basic settings**

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. |
Note that when a Job contains the parent Job and the child Job, **Component List** presents only the connection components in the same Job level.

| **Host Name** | Enter the host name or the IP address of the host server. |
| **Server Port** | Type in the listening port number of the Palo server. |
| **Username and Password** | Enter the Palo user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Database** | Type in the name of the database whose cube details you want to retrieve. |

### Advanced settings

| **tStat Catcher Statistics** | Select this check box to collect log data at the component level. |

### Global Variables

| **Global Variables** | **NB_CUBES**: the number of cubes. This is an After variable and it returns an integer.  
**CUBEID**: the ID of the cube. This is a Flow variable and it returns an integer.  
**CUBENAME**: the name of the cube. This is a Flow variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see **Talend Studio User Guide**. |

### Usage

| **Usage rule** | This component can be used as a start component. It requires an output component. |
| **Connections** | Outgoing links (from this component to another):  
**Row**: Main, Iterate; |
### Scenario: Retrieving detailed cube information from a given database

The Job in this scenario retrieves detailed information of the cubes pertaining to the demo Palo database, Biker.

To replicate this scenario, proceed as follows:

**Setting up the Job**

**Procedure**

1. Drop tPaloCubeList and tLogRow from the component Palette onto the design workspace.
2. Right-click tPaloCubeList to open the contextual menu.
3. From this menu, select Row > Main to link the two components.

**Configuring the tPaloCube component**

**Procedure**

1. Double-click the tPaloCube component to open its Component view.
2. In the **Host name** field, type in the host name or the IP address of the host server, *localhost* for this example.

3. In the **Server Port** field, type in the listening port number of the Palo server. In this scenario, it is **7777**.

4. In the **Username** field and the **Password** field, type in the authentication information. In this example, both of them are *admin*.

5. In the **Database** field, type in the database name in which you want to create the cube, *Biker* in this example.

**Job execution**

Press **F6** to run the Job.

The cube details are retrieved from the *Biker* database and are listed in the console of the **Run** view.

```
[statistics] connected
25|Orders|8|45291236380440|133605|1|0
27|bikerTalead|2|1830|2|1|0
1|GROUP_CUBE_DATA|2|8|0|1|1
2|Configuration|1|2|2|1|1
2|Configuration|1|2|2|1|1
2|Configuration|1|2|2|1|1
6|SUBSET_LOCAL|3|0|0|1|1
6|SUBSET_LOCAL|3|0|0|1|1
7|SUBSET_GLOBAL|2|0|0|1|1
7|SUBSET_GLOBAL|2|0|0|1|1
8|VIEW_LOCAL|3|0|0|1|1
8|VIEW_LOCAL|3|0|0|1|1
9|VIEW_GLOBAL|2|0|0|1|1
9|VIEW_GLOBAL|2|0|0|1|1
11|GROUP_DIMENSION_DATA_Products|2|1352|0|1|1
11|GROUP_DIMENSION_DATA_Customers|2|1864|0|1|1
```

For further information about how to interpret the cube details listed in the console, see Discovering the read-only output schema of tPaloCubeList on page 2827.
tPaloDatabase

Manages the databases inside a Palo server.

The tPaloDatabase creates, drops or recreates databases in a given Palo server.

tPaloDatabase Standard properties

These properties are used to configure tPaloDatabase running in the Standard Job framework.

The Standard tPaloDatabase component belongs to the Business Intelligence family.

The component in this framework is generally available.

Basic settings

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. 
Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Name</td>
<td>Enter the host name or the IP address of the host server.</td>
</tr>
<tr>
<td>Server Port</td>
<td>Type in the listening port number of the Palo server.</td>
</tr>
</tbody>
</table>
| Username and Password     | Enter the Palo user authentication data. 
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Database                  | Type in the name of the database on which the given operation should take place. |
| Action on database        | Select the operation you want to perform on the database of interest: 
- Create database: the database does not exist and will be created. 
- Create database if not exists: the database is created when it does not exist. 
- Delete database if exists and create: the database is deleted if exist and a new one is then created. 
- Delete database: the database is removed from the server |

Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |
Global Variables

**DATABASENAME**: the name of the database. This is an After variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](https://help.talend.com).

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used in standalone for database management in a Palo server.</th>
</tr>
</thead>
</table>

Connections

<table>
<thead>
<tr>
<th>Outgoing links (from this component to another):</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trigger</strong>: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Incoming links (from one component to this one):</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row</strong>: Iterate</td>
</tr>
<tr>
<td><strong>Trigger</strong>: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error</td>
</tr>
</tbody>
</table>

For further information regarding connections, see [Talend Studio User Guide](https://help.talend.com).

Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Scenario: Creating a database

The Job in this scenario creates a new database on a given Palo server.
To replicate this scenario, proceed as follows:

**Procedure**

1. Drop tPaloDatabase from the component Palette onto the design workspace.
2. Double-click the tPaloDatabase component to open its Component view.
3. In the **Host name** field, type in the host name or the IP address of the host server, *localhost* for this example.
4. In the **Server Port** field, type in the listening port number of the Palo server. In this scenario, it is *7777*.
5. In the **Username** field and the **Password** field, type in the authentication information. In this example, both of them are *admin*.
6. In the **Database** field, type in the database name in which you want to create the cube, *talenddatabase* in this example.
7. In the **Action on database** field, select the action to be performed. In this scenario, select *Create database* as the database to be created does not exist.
8. Press F6 to run the Job.

**Results**

A new database is created on the given Palo server.
tPaloDatabaseList

Lists database names, database types, number of cubes, number of dimensions, database status and database id from a given Palo server.

The tPaloDatabaseList retrieves a list of database details from the given Palo server.

Discovering the read-only output schema of tPaloDatabaseList

The below table presents information related to the read-only output schema of the tPaloDatabaseList component.

<table>
<thead>
<tr>
<th>Database</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database_id</td>
<td>long</td>
<td>Internal ID of the database.</td>
</tr>
<tr>
<td>Database_name</td>
<td>string</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Database_dimensions</td>
<td>int</td>
<td>Number of dimensions inside the database.</td>
</tr>
<tr>
<td>Database_cubes</td>
<td>int</td>
<td>Number of cubes inside the database.</td>
</tr>
<tr>
<td>Database_status</td>
<td>int</td>
<td>Status of the database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0 = unloaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1 = loaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2 = changed</td>
</tr>
<tr>
<td>Database_types</td>
<td>int</td>
<td>Type of the database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0 = normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1 = system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3 = user info</td>
</tr>
</tbody>
</table>

tPaloDatabaseList Standard properties

These properties are used to configure tPaloDatabaseList running in the Standard Job framework.

The Standard tPaloDatabaseList component belongs to the Business Intelligence family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.</td>
</tr>
<tr>
<td>Host Name</td>
<td>Enter the host name or the IP address of the host server.</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Server Port</td>
<td>Type in the listening port number of the Palo server.</td>
</tr>
</tbody>
</table>
| Username and Password | Enter the Palo user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | **NB_DATABASES**: the number of databases. This is an After variable and it returns an integer.  
**DATABASEID**: the ID of the database. This is a Flow variable and it returns a long.  
**DATABASENAME**: the name of the database. This is a Flow variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

**Usage**

| Usage rule | This component can be used as a start component. It requires an output component. |

| Connections | Outgoing links (from this component to another):  
**Row**: Main; Iterate  
**Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.  
Incoming links (from one component to this one):  
**Row**: Iterate |
Scenario: Retrieving detailed database information from a given Palo server

The Job in this scenario retrieves details of all of the databases from a given Palo server.

To replicate this scenario, proceed as follows:

**Setting up the Job**

*Procedure*

1. Drop `tPaloDatabaseList` and `tLogRow` from the component Palette onto the design workspace.
2. Right-click `tPaloDatabaseList` to open the contextual menu.
3. From this menu, select `Row > Main` to link the two components.

**Configuring the tPaloDatabaseList component**

*Procedure*

1. Double-click the `tPaloDatabaseList` component to open its Component view.
2. In the Host name field, type in the host name or the IP address of the host server, localhost for this example.

3. In the Server Port field, type in the listening port number of the Palo server. In this scenario, it is 7777.

4. In the Username field and the Password field, type in the authentication information. In this example, both of them are admin.

**Job execution**

Press F6 to run the Job.

Details of all of the databases in the Palo server are retrieved and listed in the console of the Run view.

```
0|System|7|6|1|1
1|Demo|23|23|1|0
6|Biker|27|28|1|0
9|tx_g|17|17|1|0
25|DBHSE24|13|12|1|0
31|elmCreate|13|12|1|0
32|tPaloOutputMulit|15|15|1|0
```

For further information about the output schema, see Discovering the read-only output schema of tPaloDatabaseList on page 2834.
tPaloDimension

Manages Palo dimensions, even elements inside a database.
The tPaloDimension creates, drops or recreates dimensions with or without dimension elements inside a Palo database.

tPaloDimension Standard properties

These properties are used to configure tPaloDimension running in the Standard Job framework.
The Standard tPaloDimension component belongs to the Business Intelligence family.
The component in this framework is generally available.

Basic settings

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level. |
| Host Name | Enter the host name or the IP address of the host server. |
| Server Port | Type in the listening port number of the Palo server. |
| Username and Password | Enter the Palo user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Database | Type in the name of the database in which the dimensions are managed. |
| Dimension | Type in the name of the dimension on which the given operation should take place. |
| Action on dimension | Select the operation you want to perform on the dimension of interest:  
- None: no action is taken on this dimension.  
- Create dimension: the dimension does not exist and will be created.  
- Create dimension if not exists: this dimension is created only when it does not exist.  
- Delete dimension if exists and create: this dimension is deleted if exist and then a new one will be created.  
- Delete dimension: this dimension is removed from the database. |
**Create dimension elements**

Select this check box to activate the dimension management fields and create dimension elements along with the creation of this dimension.

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | **DIMENSIONNAME**: the name of the dimension. This is an After variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |

**Usage**

| Usage rule | This component can be used in standalone or as end component of a process. |

| **Note:**  
The below fields are available only when the **Create dimension elements** check box is selected | **Dimension type**  
| **Note:**  
Available only when the action on dimension is **None**. |

| Commit size |
| Schema and Edit Schema |

| **Consolidation type** - **None**  
| **Note:**  
With this option, you activate the corresponding parameter fields to be completed. |
Consolidation type - Normal

**Note:**
With this option, you activate the corresponding parameter fields to be completed.

---

Consolidation type - Self-referenced

**Note:**
With this option, you activate the corresponding parameter fields to be completed.

---

Element’s type

Creation mode

---

**Connections**

Outgoing links (from this component to another):

- **Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.

Incoming links (from one component to this one):

- **Row**: Main; Iterate
- **Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.

For further information regarding connections, see *Talend Studio User Guide*.

---

**Limitation**

Deletion of dimension elements is only possible with the consolidation type **None**. Only consolidation type **Self-Referenced** allows the placing of an factor on this consolidation.

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find...
Scenario: Creating a dimension with elements

The Job in this scenario creates a date dimension with simple element hierarchy composed of three levels: Year, Month, Date.

To replicate this scenario, proceed as follows:

Setting up the Job

Procedure

1. Drop tPaloConnection, tRowGenerator, tMap, tPaloDimension from the component Palette onto the design workspace.
2. Right-click tPaloConnection to open the contextual menu and select Trigger > On Subjob Ok to link it to tRowGenerator.
3. Right-click tRowGenerator to open the contextual menu and select Row > Main to link it to tMap.
4. Right-click tMap to open the contextual menu and select Row > New output to link to tPaloDimension, then name it as out1 in the dialog box that pops up.

Setting up the DB connection

Procedure

1. Double-click the tPaloConnection component to open its Component view.
2. In the **Host name** field, type in the host name or the IP address of the host server, `localhost` for this example.

3. In the **Server Port** field, type in the listening port number of the Palo server. In this scenario, it is `7777`.

4. In the **Username** field and the **Password** field, type in the authentication information. In this example, both of them are `admin`.

**Configuring the input component**

**Procedure**

1. Double-click `tRowGenerator` to open its editor.

![Table]

2. On the upper part of the editor, click the plus button to add one column and rename it as `random_date` in the **Column** column.

3. In the newly added row, select `Date` in the **Type** column and `getRandomDate` in the **Functions** column.

4. In the **Function parameters** view on the lower part of this editor, type in the new minimum date and maximum date values in the **Value** column. In this example, the minimum is `2010-01-01`, the maximum is `2010-12-31`.

5. Click **OK** to validate your modifications and close the editor.

6. On the dialog box that pops up, click **OK** to propagate your changes.

**Configuration in the tMap editor**

**Procedure**

1. Double-click `tMap` to open its editor.
2. On the Schema editor view on the lower part of the tMap editor, under the out1 table, click the plus button to add three rows.

3. In the Column column of the out1 table, type in the new names for the three newly added rows. They are Year, Month, and Date. These rows are then added automatically into the out1 table on the upper part of the tMap editor.

4. In the out1 table on the upper part of the tMap editor, click the Expression column in the Year row to locate the cursor.

5. Press Ctrl+space to open the drop-down variable list.

6. Double-click TalendDate.formatDate to select it from the list. The expression to get the date displays in the Year row under the Expression column. The expression is TalendDate.formatDate("yyyy-MM-dd HH:mm:ss",myDate).

7. Replace the default expression with TalendDate.formatDate("yyyy",row1.random_date).

8. Do the same for the Month row and the Date row to add this default expression and to replace it with TalendDate.formatDate("MM",row1.random_date) for the Month row and with TalendDate.formatDate("dd-MM-yyyy", row1.random_date) for the Date row.

9. Click OK to validate this modification and accept the propagation by clicking OK in the dialog box that pops up.

Configuring the tPaloDimension component

Procedure

1. On the workspace, double-click tPaloDimension to open its Component view.
2. Select the **Use an existing connection** check box. Then **tPaloConnection_1** displays automatically in the **Connection configuration** field.

3. In the **Database** field, type in the database in which the new dimension is created, **talendDatabase** for this scenario.

4. In the **Dimension** field, type in the name you want to use for the dimension to be created, for example, **Date**.

5. In the **Action on dimension** field, select the action to be performed. In this scenario, select **Create dimension if not exist**.

6. Select the **Create dimension elements** check box.

7. In the **Consolidation Type** area, select the **Normal** check box.

8. Under the element hierarchy table in the **Consolidation Type** area, click the plus button to add three rows into the table.

9. In the **Input column** column of the element hierarchy table, select **Year** from the drop-down list for the first row, **Month** for the second and **Date** for the third. This determinates levels of elements from different columns of the input schema.

**Job execution**

Press F6 to run the Job.

A new dimension is then created in your Palo database **talendDatabase**.
tPaloDimensionList

Retrieves a list of dimension details from the given Palo database.

tPaloDimensionList lists dimension names, dimension types, number of dimension elements, maximum dimension indent, maximum dimension depth, maximum dimension level, dimension id from a given Palo server.

Discovering the read-only output schema of tPaloDimensionList

The below table presents information related to the read-only output schema of the tPaloDimensionList component.

<table>
<thead>
<tr>
<th>Database</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension_id</td>
<td>long</td>
<td>Internal ID of the dimension.</td>
</tr>
<tr>
<td>Dimension_name</td>
<td>string</td>
<td>Name of the dimension.</td>
</tr>
<tr>
<td>Dimension_attribute_cube</td>
<td>string</td>
<td>Name of the cube of attributes.</td>
</tr>
<tr>
<td>Dimension_rights_cube</td>
<td>string</td>
<td>Name of the cube of rights.</td>
</tr>
<tr>
<td>Dimension_elements</td>
<td>int</td>
<td>Number of the dimension elements</td>
</tr>
<tr>
<td>Dimension_max_level</td>
<td>int</td>
<td>Maximum level of the dimension</td>
</tr>
<tr>
<td>Dimension_max_indent</td>
<td>int</td>
<td>Maximum indent of the dimension</td>
</tr>
<tr>
<td>Dimension_max_depth</td>
<td>int</td>
<td>Maximum depth of the dimension</td>
</tr>
<tr>
<td>Dimension_type</td>
<td>int</td>
<td>Type of the dimension.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0 =normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1 =system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2 =attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3 =user info</td>
</tr>
</tbody>
</table>

tPaloDimensionList Standard properties

These properties are used to configure tPaloDimensionList running in the Standard Job framework.
The Standard tPaloDimensionList component belongs to the Business Intelligence family.
The component in this framework is generally available.

Basic settings

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. |
Note that when a Job contains the parent Job and the child Job, **Component List** presents only the connection components in the same Job level.

| **Host Name** | Enter the host name or the IP address of the host server. |
| **Server Port** | Type in the listening port number of the Palo server. |
| **Username and Password** | Enter the Palo user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| **Database** | The name of the database where the dimensions of interest reside. |
| **Retrieve cube dimensions** | Select this check box to retrieve dimension information from an existing cube. |
| **Cube** | Type in the name of the cube from which dimension information is retrieved. **Note:** Available when you select the **Retrieve cube dimensions** check box. |
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.  
  
  **Built-in**: The schema is created and stored locally for this component only. Related topic: see **Talend Studio User Guide**.  
  
  **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see **Talend Studio User Guide**. |
## Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

## Global Variables

### Global Variables

<table>
<thead>
<tr>
<th>DIMENSIONNAME</th>
<th>the name of the dimension. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

## Usage

### Usage rule

This component can be used in standalone or as start component of a process.

### Connections

Outgoing links (from this component to another):

- **Row**: Main; Iterate.
- **Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.

Incoming links (from one component to this one):

- **Row**: Iterate
- **Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.

For further information regarding connections, see *Talend Studio User Guide*.

### Limitation

The output schema is fixed and read-only.

Due to license incompatibility, one or more JARs required to use this component are not provided.

You can install the missing JARs for this particular component by clicking the **Install** button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).
**Scenario: Retrieving detailed dimension information from a given database**

The Job in this scenario retrieves details of all of the dimensions from a given database.

To replicate this scenario, proceed as follows:

**Setting up the Job**

**Procedure**

1. Drop tPaloDimensionList and tLogRow from the component Palette onto the design workspace.
2. Right-click tPaloDimensionList to open the contextual menu.
3. From this menu, select Row > Main to link the two components.

**Configuring the tPaloDimensionList component**

**Procedure**

1. Double-click the tPaloDimensionList component to open its Component view.

2. In the **Host name** field, type in the host name or the IP address of the host server, *localhost* for this example.
3. In the **Server Port** field, type in the listening port number of the Palo server. In this scenario, it is *7777*.
4. In the **Username** field and the **Password** field, type in the authentication information. In this example, both of them are *admin*.
5. In the **Database** field, type in the database name where the dimensions of interest reside, *Biker* in this example.
Job execution

Press **F6** to run the Job.

Details of all the dimensions in the *Biker* database are retrieved and listed in the console of the **Run** view.

For further information about the output schema, see Discovering the read-only output schema of **tPaloDimensionList** on page 2846.
tPaloInputMulti

Retrieves the stored or calculated values in combination with the element records out of a cube.
The tPaloInputMulti retrieves data (elements as well as values) from a Palo cube.

tPaloInputMulti Standard properties

These properties are used to configure tPaloInputMulti running in the Standard Job framework.
The Standard tPaloInputMulti component belongs to the Business Intelligence family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.</td>
</tr>
<tr>
<td>Host Name</td>
<td>Enter the host name or the IP address of the host server.</td>
</tr>
<tr>
<td>Server Port</td>
<td>Type in the listening port number of the Palo server.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Enter the Palo user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Database</td>
<td>Type in the name of the database where the elements of interest reside.</td>
</tr>
<tr>
<td>Cube</td>
<td>Type in the name of the cube where the dimension elements to be retrieved are stored.</td>
</tr>
<tr>
<td>Cube type</td>
<td>Select the cube type from the drop-down list for the cube of concern. This type may be: Normal, Attribut, System, User Info</td>
</tr>
<tr>
<td>Commit size</td>
<td>Type in the row count of each batch to be retrieved.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
</tbody>
</table>
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

**Cube Query**  
Complete this table with the query you want to use to retrieve data. The columns to be filled are:

- **Column**: the schema columns are added automatically to this column once defined in the schema editor. The schema columns are used to store the retrieved dimension elements.

- **Dimensions**: type in each of the dimension names of the cube from which you want to retrieve dimension elements.

  **Warning:**  
  *The dimension order listed in this column must be consistent with the order given in the cube that stores these dimensions.*

- **Elements**: type in the dimension elements from which data is retrieved. If several elements are needed from one single dimension, separate them with a comma.

**Advanced settings**

**tStat Catcher Statistics**  
Select this check box to collect log data at the component level.

**Global Variables**

**Global Variables**  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions...
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component requires an output component.</th>
</tr>
</thead>
</table>

#### Connections

Outgoing links (from this component to another):

- **Row**: Main
- **Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.

Incoming links (from one component to this one):

- **Row**: Iterate
- **Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.

For further information regarding connections, see *Talend Studio User Guide*.

### Limitation

According to the architecture of OLAP-Systems only one single value (text or numeric) could be retrieved from the cube. The **MEASURE** column and the **TEXT** column are fixed and read-only.

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

### Scenario: Retrieving dimension elements from a given cube

The Job in this scenario retrieves several dimension elements from a demo Palo cube *Sales*.
To replicate this scenario, proceed as follows:

**Setting up the Job**

**Procedure**

1. Drop `tPaloInputMulti` and `tLogRow` from the component Palette onto the design workspace.
2. Right-click `tPaloInputMulti` to open its contextual menu.
3. In the menu, select Row > Main to connect `tPaloInputMulti` to `tLogRow` with a row link.

**Setting up the DB connection**

**Procedure**

1. Double-click the `tPaloInputMulti` component to open its Component view.

   ![Component view](image)

   - **Host name** field: Type in the host name or the IP address of the host server, localhost for this example.
   - **Server Port** field: Type in the listening port number of the Palo server. In this scenario, it is 7777.
   - **Username** field and the **Password** field: Type in the authentication information. In this example, both of them are admin.

**Configuring the Cube Query**

**Procedure**

1. In the **Database** field, type in the database name in which the cube to be used is stored.
2. In the **Cube** field, type in the cube name in which the dimensions of interests are stored. In this scenario, it is one of the demo cubes *Sales*.

3. In the **Cube type** field, select the **Normal** type from the drop-down list for the cube to be created, meaning this cube will be normal and default.

4. Next to the **Edit schema** field, click the three-dot button to open the schema editor.

5. In the schema editor, click the plus button to add the rows of the schema to be edited. In this example, add rows corresponding to all of the dimensions stored in the *Sales* cube: **Products, Regions, Months, Years, Datatypes, Measures**. Type in them in the order given in this cube.

6. Click **OK** to validate this editing and accept the propagation of this change to the next component. Then these columns are added automatically into the **Column** column of the **Cube query** table in the **Component** view. If the order is not consistent with the one in the *Sales* cube, adapt it using the up and down arrows under the schema table.

7. In the **Dimensions** column of the **Cube query** table, type in each of the dimension names stored in the *Sales* cube regarding to each row in the **Column** column. In the *Sales* cube, the dimension names are: **Products, Regions, Months, Years, Datatypes, Measures**.

8. In the **Elements** columns of the **Cube query** table, type in the dimension elements you want to retrieve regarding to the dimensions they belong to. In this example, the elements to be retrieved are **All Products, Germany and Austria** (Belonging to the same dimension **Regions**, these two elements are entered in the same row and separated with a comma), **Jan, 2009, Actual, Turnover**.

**Job execution**

**Procedure**

1. Click **tLogRow** to open its **Component** view.
2. In the **Mode** area, select the **Table (print values in cells of a table)** check box to display the execution result in a table.

3. Press **F6** to run the Job.

**Results**

The dimension elements and the corresponding Measure values display in the **Run** console.

```
[statistics] connecting to socket on port 3338
[statistics] connected

<table>
<thead>
<tr>
<th>Products</th>
<th>Regions</th>
<th>Months</th>
<th>Years</th>
<th>Datatypes</th>
<th>Measures</th>
<th>MEASURE</th>
<th>TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Products</td>
<td>Austria</td>
<td>Jan</td>
<td>2009</td>
<td>Actual</td>
<td>Turnover</td>
<td>476,977.95</td>
<td>null</td>
</tr>
<tr>
<td>All Products</td>
<td>Germany</td>
<td>Jan</td>
<td>2009</td>
<td>Actual</td>
<td>Turnover</td>
<td>1234.56</td>
<td>null</td>
</tr>
</tbody>
</table>

[statistics] disconnected
Job retrieve ended at 13:49 02/11/2010. (exit code=0)
```
**tPaloOutput**

Takes the input stream and writes it to a given Palo cube. The tPaloOutput writes one row of data (elements as well as values) into a Palo cube.

**tPaloOutput Standard properties**

These properties are used to configure tPaloOutput running in the Standard Job framework. The Standard tPaloOutput component belongs to the Business Intelligence family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.</td>
</tr>
<tr>
<td>Host Name</td>
<td>Enter the host name or the IP address of the host server.</td>
</tr>
<tr>
<td>Server Port</td>
<td>Type in the listening port number of the Palo server.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Enter the Palo user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Database</td>
<td>Type in the name of the database where the cube of interest resides.</td>
</tr>
<tr>
<td>Cube</td>
<td>Type in the name of the cube in which the incoming data is written.</td>
</tr>
<tr>
<td>Commit size</td>
<td>Type in the row count of each batch to be written into the cube.</td>
</tr>
</tbody>
</table>
| Schema and Edit Schema   | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  - View schema: choose this option to view the schema only.  
  - Change to built-in property: choose this option to change the schema to Built-in for local changes. |
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

| Built-in | The schema is created and stored locally for this component only. Related topic: see [Talend Studio User Guide](#). |
| Repository | The schema already exists and is stored in the Repository, hence can be reused. Related topic: see [Talend Studio User Guide](#). |

**Column as Measure**
Select the column from the input stream which holds the Measure or Text values.

**Create element if not exist**
Select this check box to create the element being processed if it does not exist originally.

**Save cube at process end**
Select this check box to save the cube you have written the data in at the end of this process.

### Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

### Global variables

| Global variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
 **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
 A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
 To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
 For further information about variables, see [Talend Studio User Guide](#). |

### Usage

| Usage rule | This component requires an input component. |
### Connections

Outgoing links (from this component to another):
- **Row**: Iterate  
- **Trigger**: Run if  

Incoming links (from one component to this one):
- **Row**: Main; Reject  

For further information regarding connections, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component is able to write only one row of data into a cube. Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the <strong>Install</strong> button on the <strong>Component</strong> tab view. You can also find out and add all missing JARs easily on the <strong>Modules</strong> tab in the <strong>Integration</strong> perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

### Related scenario

For related topic, see *Scenario 1: Writing data into a given cube* on page 2862.
tPaloOutputMulti

Takes the input stream and writes it to a given Palo cube.

The tPaloOutputMulti writes data (elements as well as values) into a Palo cube.

tPaloOutputMulti Standard properties

These properties are used to configure tPaloOutputMulti running in the Standard Job framework. The Standard tPaloOutputMulti component belongs to the Business Intelligence family. The component in this framework is generally available.

Basic settings

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level. |
| Host Name | Enter the host name or the IP address of the host server. |
| Server Port | Type in the listening port number of the Palo server. |
| Username and Password | Enter the Palo user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Database | Type in the name of the database where the cube of interest resides. |
| Cube | Type in the name of the cube in which the incoming data is written. |
| Cube type | Select the cube type from the drop-down list for the cube of concern. This type may be: Normal, Attribut, System, User Info |
| Commit size | Type in the row count of each batch to be written into the cube. |
| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. |
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

| **Built-in**: The schema is created and stored locally for this component only. Related topic: see **Talend Studio User Guide**. |
| **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see **Talend Studio User Guide**. |

**Measure value**
Select the column from the input stream which holds the Measure or Text values.

**Splash mode**
Select the splash mode used to write data into a consolidated element. The mode may be:

- **Add**: it writes values to the underlying elements.
- **Default**: it uses the default splash mode.
- **Set**: it simply sets or replaces the current value and make the distribution based on the other values.
- **Disable**: it applies no splashing.

For further information about the Palo splash modes, see Palo’s user guide.

**Add values**
Select this check box to add new values to the current values for a sum. Otherwise these new values will overwrite the current ones.

**Use eventprocessor**
Select this check box to call the supervision server.

**Die on error**
This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

**Advanced settings**

| **tStat Catcher Statistics** | Select this check box to collect log data at the component level. |
### Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](https://help.talend.com). |

### Usage

| Usage rule | This component requires an input component. |

| Connections | Outgoing links (from this component to another):

**Row**: Main

**Trigger**: Run if; On Component Ok; On Component Error.

Incoming links (from one component to this one):

**Row**: Main; Reject

For further information regarding connections, see [Talend Studio User Guide](https://help.talend.com). |

| Limitation | Numeric measures are only be accepted as Double or String type. When the string type is used, write the value to be processed between quotation marks.

Due to license incompatibility, one or more JARs required to use this component are not provided.

You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |

### Scenario 1: Writing data into a given cube

The Job in this scenario writes new values in the **Sales** cube given as demo in the **Demo** database installed with Palo.
To replicate this scenario, proceed as follows:

**Setting up the Job**

**Procedure**

1. Drop `tFixedFlowInput` and `tPaloOutputMulti` from the component Palette onto the design workspace.
2. Right-click `tFixedFlowInput` to open its contextual menu.
3. In this menu, select **Row > Main** to connect this component to `tPaloOutputMulti`.

**Configuring the input component**

**Procedure**

1. Double-click the `tFixedFlowInput` component to open its **Component** view.

![Component view](image)

- **Basic settings**
- **Advanced settings**
- **Dynamic settings**
- **View**
- **Documentation**

2. Click the three-dot button to open the schema editor.

![Schema editor](image)
3. In the schema editor, click the plus button to add 7 rows and rename them respectively as Products, Regions, Months, Years, Datatypes, Measures and Values. The order of these rows must be consistent with that of the corresponding dimensions in the Sales cube and the type of the Value column where the measure value resides is set to double/Double.

4. Click OK to validate the editing and accept the propagation prompted by the dialog box that pops up. Then the schema column labels display automatically in the Value table under the Use single table check box, in the Mode area.

5. In the Value table, type in values for each row in the Value column. In this example, these values are: Desktop L, Germany, Jan, 2009, Actual, Turnover, 1234.56.

**Configuring the output component**

**Procedure**

1. Double-click tPaloOutputMulti to open its Component view.
2. In the **Server Port** field, type in the listening port number of the Palo server. In this scenario, it is 7777.

3. In the **Username** field and the **Password** field, type in the authentication information. In this example, both of them are `admin`.

4. In the **Database** field, type in the database name in which you want to create the cube, *Demo* in this example.

5. In the **Cube** field, type in the name of the cube you want to write data in, for example, *Sales*.

6. In the **Cube type** field, select the **Normal** type from the drop-down list for the cube to be created, meaning this cube will be normal and default.

7. In the **Measure Value** field, select the Measure element. In this scenario, select *Value*.

**Job execution**

Press **F6** to run the Job.

The inflow data has been written into the *Sales* cube.

---

**Scenario 2: Rejecting inflow data when the elements to be written do not exist in a given cube**

The Job in this scenario tries to write data into the *Sales* cube but as the elements of interest do not exist in this cube, the inflow data is rejected.

---

To replicate this scenario, proceed as follows:
Setting up the Job

Procedure

1. Drop **tFixedFlowInput**, **tPaloCheckElements**, **tPaloOutputMulti** and **tLogRow** from the component Palette onto the design workspace.
2. Right-click **tFixedFlowInput** to open its contextual menu.
3. In this menu, select **Row > Main** to connect this component to **tPaloCheckElements**.
4. Do the same to connect **tPaloOutputMulti** using **row** link.
5. Right-click **tPaloCheckElements** to open its contextual menu.
6. In this menu, select **Row > Reject** to connect this component to **tLogRow**.

Configuring the input component

Procedure

1. Double-click the **tFixedFlowInput** component to open its Component view.
2. Click the three-dot button to open the schema editor.

2. Click the three-dot button to open the schema editor.
3. In the schema editor, click the plus button to add 7 rows and rename them respectively as Products, Regions, Months, Years, Datatypes, Measures and Values. The order of these rows must be consistent with that of the corresponding dimensions in the Sales cube and the type of the Value column where the measure value resides is set to double/Double.

4. Click OK to validate the editing and accept the propagation prompted by the dialog box that pops up. Then the schema column labels display automatically in the Value table under the Use single table check box, in the Mode area.

5. In the Value table, type in values for each row in the Value column. In this example, these values are: Smart Products, Germany, Jan, 2009, Actual, Turnover, 1234.56. The Smart Products element does not exist in the Sales cube.

**Configuring the tPaloCheckElements component**

**Procedure**

1. Double-click tPaloCheckElements to open its Component view.
2. In the **Host name** field, type in *localhost*.

3. In the **Server Port** field, type in the listening port number of the Palo server. In this scenario, it is **7777**.

4. In the **Username** field and the **Password** field, type in the authentication information. In this example, both of them are *admin*.

5. In the **Database** field, type in the database name in which you want to create the cube, *Demo* in this example.

6. In the **Cube** field, type in the name of the cube you want to write data in, for example, *Sales*.

7. In the **On Element error** field, select **Reject row** from the drop-down list.

8. In the element table at the bottom of the **Basic settings** view, click the **Element type** column in the **Value** row and select **Measure** from the drop down list.

### Configuring the output component

**Procedure**

1. Double-click **tPaloOutputMulti** to open its **Component** view.
2. In the **Server Port** field, type in the listening port number of the Palo server. In this scenario, it is 7777.

3. In the **Username** field and the **Password** field, type in the authentication information. In this example, both of them are `admin`.

4. In the **Database** field, type in the database name in which you want to create the cube, `Demo` in this example.

5. In the **Cube** field, type in the name of the cube you want to write data in, for example, `Sales`.

6. In the **Cube type** field, select the **Normal** type from the drop-down list for the cube to be created, meaning this cube will be normal and default.

7. In the **Measure Value** field, select the Measure element. In this scenario, select `Value`.

### Job execution

Press **F6** to run the Job.

The data to be written is rejected and displayed in the console of the Run view. You can read that the error message is **Smart Products**.

```
[statistics] connecting to socket on port 3407
[statistics] connected

+-------------------------+-----------------------------+-----------------+-----------+---------------------+----------+----------+-----------------------------+-------------+-----------+-----------------------------+----------
| Products | Regions | Months | Years | Datatypes | Measures | Value | errorMessage |
|-----------|---------|--------|-------|-----------|---------|-------|---------------|-------------|-----------|---------------|---------|
| Smart Products| Germany | Jan | 2009 | | | | |
| 1234.56 | Products: Smart Products |

[statistics] disconnected
Job JobOutputExistingElements ended at 14:45 09/11/2010. [exit code=0]```
tPaloRule

Manages rules in a given cube.

The tPaloRule creates or modifies rules in a given cube.

**tPaloRule Standard properties**

These properties are used to configure tPaloRule running in the Standard Job framework.

The Standard tPaloRule component belongs to the Business Intelligence family.

The component in this framework is generally available.

**Basic settings**

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.
Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level. |
| Host Name | Enter the host name or the IP address of the host server. |
| Server Port | Type in the listening port number of the Palo server. |
| Username and Password | Enter the Palo user authentication data.
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Database | Type in the name of the database where the dimensions applying the rules of interest reside. |
| Cube | Type in the name of the cube whose dimension information is retrieved. |
| Cube rules | Complete this table to perform various actions on specific rules.
**Definition:** type in the rule to be applied.**
**External Id:** type in the user-defined external ID.**
**Comment:** type in comment for this rule.**
**Activated:** select this check box to activate this rule.**
**Action:** select the action to be performed from the drop-down list.
- **Create:** create this rule.
- **Delete:** delete this rule. |
**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. |
| | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
| | To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. |
| | For further information about variables, see **Talend Studio User Guide**. |

**Usage**

| Usage rule | This component can be used in standalone for rule creation, deletion or update. |

**Connections**

| Outgoing links (from this component to another): |
| **Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error. |

| Incoming links (from one component to this one): |
| **Row**: Iterate |
| **Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error. |

For further information regarding connections, see **Talend Studio User Guide**.

**Limitation**

| Update or deletion of a rule is available only when this rule has been created with external ID. |
| Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in **Talend Help Center** (https://help.talend.com). |
Scenario: Creating a rule in a given cube

The Job in this scenario creates a rule applied on dimensions of a given cube.

To replicate this scenario, proceed as follows:

**Setting up the DB connection**

**Procedure**

1. Drop **tPaloRule** from the component **Palette** onto the design workspace.
2. Double-click the **tPaloRule** component to open its **Component** view.
3. In the **Host name** field, type in the host name or the IP address of the host server, *localhost* for this example.
4. In the **Server Port** field, type in the listening port number of the Palo server. In this scenario, it is 7777.
5. In the **Username** field and the **Password** field, type in the authentication information. In this example, both of them are *admin*.
6. In the **Database** field, type in the database name in which the dimensions applying the created rules reside, *Biker* in this example.
7. In the **Cube** field, type in the name of the cube which the dimensions applying the created rules belong to, for example, *Orders*.

### Setting the Cube rules

#### Procedure

1. Under the **Cube rules** table, click the plus button to add a new row.
2. In the **Cube rules** table, type in `['2009'] = 123` in the **Definition** column, `OrderRule1` in the **External Id** column and *Palo Demo Rules* in the **Comment** column.
3. In the **Activated** column, select the check box.
4. In the **Action** column, select **Create** from the drop-down list.

#### Job execution

Press **F6** to run the Job.

The new rule has been created and the value of every `2009` element is **123**.
tPaloRuleList

Lists all rules, formulas, comments, activation status, external IDs from a given cube. The tPaloRuleList retrieves a list of rule details from the given Palo database.

Discovering the read-only output schema of tPaloRuleList

The following table presents information related to the read-only output schema of the tPaloRuleList component.

<table>
<thead>
<tr>
<th>Database</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rule_identifier</td>
<td>long</td>
<td>The internal identifier/id for this rule.</td>
</tr>
<tr>
<td>rule_definition</td>
<td>string</td>
<td>The formula of this rule. For further information about this formula, see the Palo user guide.</td>
</tr>
<tr>
<td>ruleExtern_id</td>
<td>string</td>
<td>The user-defined external id.</td>
</tr>
<tr>
<td>ruleComment</td>
<td>string</td>
<td>The user-edited comment on this rule.</td>
</tr>
<tr>
<td>ruleActivated</td>
<td>boolean</td>
<td>Indicates if this rule had been activated or not.</td>
</tr>
</tbody>
</table>

tPaloRuleList Standard properties

These properties are used to configure tPaloRuleList running in the Standard Job framework. The Standard tPaloRuleList component belongs to the Business Intelligence family. The component in this framework is generally available.

Basic settings

- **Use an existing connection**: Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.
  Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.

- **Host Name**: Enter the host name or the IP address of the host server.

- **Server Port**: Type in the listening port number of the Palo server.

- **Username** and **Password**: Enter the Palo user authentication data.
  To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.
### Database
The name of the database where the cube of interest resides.

### Cube
Type in the name of the cube in which you want to retrieve the rule information.

### Schema and Edit Schema
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

#### Built-in
The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

#### Repository
The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

### Advanced settings

<table>
<thead>
<tr>
<th>tStat Catcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_RULES: the number of rules. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EXTERNAL_RULEID: the external ID of the rule. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
</tbody>
</table>
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide.*

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used in standalone or as start component of a process.</th>
</tr>
</thead>
</table>

### Connections

Outgoing links (from this component to another):

- **Row:** Main; Iterate.
- **Trigger:** Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.

Incoming links (from one component to this one):

- **Row:** Iterate
- **Trigger:** Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.

For further information regarding connections, see *Talend Studio User Guide.*

### Limitation

The output schema is fixed and read-only.

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

### Scenario: Retrieving detailed rule information from a given cube

The Job in this scenario retrieves rule details applied on the dimensions of a given cube.

To replicate this scenario, proceed as follows:

1. Drop **tPaloRuleList** and **tLogRow** from the component **Palette** onto the design workspace.
2. Right-click tPaloRuleList to open the contextual menu.
3. From this menu, select Row > Main to link the two components.

**Configuring the tPaloRuleList component**

**Procedure**

1. Double-click the tPaloRuleList component to open its Component view.

![Component view](image)

2. In the Host name field, type in the host name or the IP address of the host server, *localhost* for this example.
3. In the Server Port field, type in the listening port number of the Palo server. In this scenario, it is 7777.
4. In the Username and Password fields, type in the authentication information. In this example, both of them are *admin*.
5. In the Database field, type in the database name where the dimensions applying the rules of interest reside, *Biker* in this example.
6. In the Cube field, type in the name of the cube which the rules of interest belong to.

**Job execution**

Press **F6** to run the Job.

Details of all of the rules in the *Orders* cube are retrieved and listed in the console of the Run view.
For further information about the output schema, see Discovering the read-only output schema of tPaloRuleList on page 2874.
tParAccelBulkExec

Improves performance when loading data in ParAccel database.

The tParAccelOutputBulk and tParAccelBulkExec are generally used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tParAccelOutputBulkExec component, detailed in a different section. The advantage of using two separate steps is that the data can be transformed before it is loaded in the database.

The tParAccelBulkExec performs an Insert action on the data.

tParAccelBulkExec Standard properties

These properties are used to configure tParAccelBulkExec running in the Standard Job framework.

The Standard tParAccelBulkExec component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>Note:</td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
<td></td>
</tr>
</tbody>
</table>
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of the DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Database name.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td>On the table defined, you can perform one of the following operations: <strong>None</strong>: No operation is carried out. <strong>Drop and create table</strong>: The table is removed and created again. <strong>Create table</strong>: The table does not exist and gets created. <strong>Create table if not exists</strong>: The table is created if it does not exist. <strong>Drop table if exists and create</strong>: The table is removed if already exists and created again. <strong>Clear table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields. <strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>. <strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>. When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.</td>
</tr>
</tbody>
</table>
You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

### Advanced settings

| Copy mode | Select the copy mode you want to use from either:
| Basic: Standard mode, without optimisation.  
Parallel: Allows you to use several internal ParAccel APIs in order to optimise loading speed. |
| Filename | Name of the file to be loaded.

⚠️ **Warning:**  
This file is located on the machine specified by the URI in the **Host** field so it should be on the same machine as the database server.

| File Type | Select the file type from the list. |
| Field Layout | Select the field layout from the list. |
| Field separator | Character, string or regular expression to separate fields. |
| Explicit IDs | The ID is already present in the file to be loaded or will be set by the database. |
| Remove Quotes | Select this check box to remove quotation marks from the file to be loaded. |
| Max. Errors | Type in the maximum number of errors before your Job stops. |
| Date Format | Type in the date format to be used. |
| Time/Timestamp Format | Enter the date and hour format to be used. |
| Additional COPY Options | Enter the specific, customized ParAccel option that you want to use. |
| Log file | Browse to or enter the access path to the log file in your directory. |
| Logging level | Select the information type you want to record in your log file. |
| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component covers all possible SQL database queries. It allows you to carry out actions on a table or on the data of a table in a ParAccel database. It enables you to create a reject flow, with a Row > Reject link filtering the data in error. For a usage example, see Scenario: Retrieving data in error with a Reject link on page 2529. |
| Dynamic settings | Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.  

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |
Related scenarios

For related scenarios, see:

- Scenario: Inserting transformed data in MySQL database on page 2537 of tMysqlOutputBulk.
- Scenario: Inserting data in bulk in MySQL database on page 2544 of tMysqlOutputBulkExec.
- Scenario: Truncating and inserting file data into an Oracle database on page 2752 of tOracleBulkExec.
tParAccelClose

Closes a transaction.
The tParAccelClose closes the transaction committed in the connected DB.

**tParAccelClose Standard properties**

These properties are used to configure tParAccelClose running in the Standard Job framework.
The Standard tParAccelClose component belongs to the Databases family.
The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tParAccelConnection</strong> component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is to be used along with ParAccel components, especially with **tParAccelConnection** and **tParAccelCommit**. |

| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable. |
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

No scenario is available for the Standard version of this component yet.
tParAccelCommit

Commits in one go a global transaction, using a unique connection, instead of doing that on every row or every batch and thus provides gain in performance.

The tParAccelCommit validates the data processed through the job into the connected DB.

**tParAccelCommit Standard properties**

These properties are used to configure tParAccelCommit running in the Standard Job framework.

The Standard tParAccelCommit component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tParAccelConnection</strong> component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task. <strong>Warning:</strong> If you want to use a <strong>Row &gt; Main</strong> connection to link <strong>tParAccelCommit</strong> to your Job, your data will be committed row by row. In this case, do not select the <strong>Close connection</strong> check box or your connection will be closed before the end of your first row commit.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tParAccel* components, especially with |
Dynamic settings

Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Related scenario

For tParAccelCommit related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476
tParAccelConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

The tParAccelConnection opens a connection to the database for a current transaction.

**tParAccelConnection Standard properties**

These properties are used to configure tParAccelConnection running in the Standard Job framework.

The Standard tParAccelConnection component belongs to the Databases and the ELT families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Name of the schema</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Use or register a shared DB Connection</strong></td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the Shared DB Connection Name field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among</td>
</tr>
</tbody>
</table>
several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the Use dynamic job and Use an independent process to run subjob options of the tRunJob component. Using a shared connection together with a tRunJob component with either of these two options enabled will cause your Job to fail.

Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Commit</td>
<td>Select this check box to commit any changes to the database automatically upon the transaction. With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component. Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component is more commonly used with other tParAccel* components, especially with the tParAccelCommit and tParAccelRollback components.</td>
<td></td>
</tr>
</tbody>
</table>

Limitation

<table>
<thead>
<tr>
<th>Limitation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
<td></td>
</tr>
</tbody>
</table>

Related scenario

For tParAccelConnection related scenario, see tMysqlConnection on page 2475
tParAccelInput

Reads a database and extracts fields based on a query.

`tParAccelInput` executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Main row link.

**tParAccelInput Standard properties**

These properties are used to configure `tParAccelInput` running in the Standard Job framework.

The `tParAccelInput` component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see **Talend Studio User Guide**.

Use an existing connection

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component.
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](#).

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of the DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Schema</td>
<td>Exact name of the schema</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
<tr>
<td></td>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
<tr>
<td></td>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the <a href="#">Repository Content</a> window.</td>
</tr>
<tr>
<td>Table name</td>
<td>Name of the table to be read.</td>
</tr>
</tbody>
</table>
Query type and Query

Enter your DB query paying particularly attention to sequence the fields properly in order to match the schema definition.

Guess Query

Click the **Guess Query** button to generate the query which corresponds to your table schema in the Query field.

Guess schema

Click the **Guess schema** button to retrieve the table schema.

Advanced settings

Use cursor

When selected, helps to decide the row set to work with at a time and thus optimize performance.

Trim all the String/Char columns

Select this check box to remove leading and trailing whitespace from all the String/Char columns.

Trim column

Remove leading and trailing whitespace from defined columns.

tStat Catcher Statistics

Select this check box to collect log data at the component level.

Global Variables

Global Variables

**NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

Usage

Usage rule

This component covers all possible SQL queries for ParAccel databases.

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular
component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

**Dynamic settings**

Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Related scenarios**

For related scenarios, see:
**tParAccelOutput**

Executes the action defined on the table and/or on the data of a table, according to the input flow from the previous component.

The tParAccelOutput writes, updates, modifies or deletes the data in a database.

**tParAccelOutput Standard properties**

These properties are used to configure tParAccelOutput running in the Standard Job framework.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

![Icon]

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component</td>
</tr>
</tbody>
</table>

2894
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of the DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Database name.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td><strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Create table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td><strong>Create table if not exists</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop table if exists and create</strong>: The table is removed if already exists and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td><strong>Action on data</strong></td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Make changes to existing entries</td>
</tr>
<tr>
<td></td>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td><strong>Update or insert</strong>: Update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Remove entries corresponding to the input flow.</td>
</tr>
</tbody>
</table>

⚠️ **Warning:**
It is necessary to specify at least one column as a primary key on which the **Update** and **Delete** operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the **Update** and **Delete** operations. To do that: Select the **Use field options** check box and then in the **Key in update column**, select the check boxes next to the column names you want to use as a base for the **Update** operation. Do the same in the **Key in delete column** for the **Delete** operation.

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <strong>Talend Studio User Guide</strong>. When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
<td></td>
</tr>
<tr>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
<td></td>
</tr>
<tr>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the <strong>[Repository Content]</strong> window.</td>
<td></td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>
## Advanced settings

<table>
<thead>
<tr>
<th><strong>Commit every</strong></th>
<th>Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional Columns</strong></td>
<td>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</td>
</tr>
<tr>
<td><strong>Name:</strong></td>
<td>Type in the name of the schema column to be altered or inserted as new column</td>
</tr>
<tr>
<td><strong>SQL expression:</strong></td>
<td>Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td><strong>Position:</strong></td>
<td>Select Before, Replace or After following the action to be performed on the reference column.</td>
</tr>
<tr>
<td><strong>Reference column:</strong></td>
<td>Type in a column of reference that the tDBOutput can use to place or replace the new or altered column.</td>
</tr>
<tr>
<td><strong>Use field options</strong></td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td><strong>Use Batch</strong></td>
<td>Select this check box to activate the batch mode for data processing.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>This check box is available only when you have selected the Insert, Update, or Delete option in the Action on data option.</td>
</tr>
<tr>
<td><strong>Batch Size</strong></td>
<td>Specify the number of records to be processed in each batch.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>NB_LINE:</strong> the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_UPDATED:</strong> the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED:</strong> the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_DELETED:</strong> the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>
**Usage**

| Usage rule | This component covers all possible SQL database queries. It allows you to carry out actions on a table or on the data of a table in a ParAccel database. It enables you to create a reject flow, with a Row > Rejects link filtering the data in error. For a usage example, see Scenario: Retrieving data in error with a Reject link on page 2529. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.  

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |
| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the |
Related scenarios

For a related scenario, see:

- Scenario: Inserting a column and altering data using tMysqlOutput on page 2521.
tParAccelOutputBulk

Prepares the file to be used as parameter in the INSERT query to feed the ParAccel database.

The tParAccelOutputBulk and tParAccelBulkExec are generally used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tParAccelOutputBulkExec component, detailed in a different section. The advantage of using two separate steps is that the data can be transformed before it is loaded in the database.

The tParAccelOutputBulk writes a file with columns based on the defined delimiter and the ParAccel standards.

**tParAccelOutputBulk Standard properties**

These properties are used to configure tParAccelOutputBulk running in the Standard Job framework.

The Standard tParAccelOutputBulk component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>File Name</td>
<td>Name of the file to be generated.</td>
</tr>
<tr>
<td><strong>Warning:</strong> This file is generated on the local machine or a shared folder on the LAN.</td>
<td></td>
</tr>
<tr>
<td>Append</td>
<td>Select this check box to add the new rows at the end of the file</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
</tbody>
</table>
**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-In** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Row separator</strong></th>
<th>String (ex: <code>\n</code> on Unix) to distinguish rows.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Field separator</strong></td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td><strong>Include header</strong></td>
<td>Select this check box to include the column header.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select the encoding type from the list or select Custom and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer. <strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions</th>
</tr>
</thead>
</table>
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with tParAccelBulkEexec component. Used together they offer gains in performance while feeding a ParAccel database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component family</td>
<td>Databases/ParAccel</td>
</tr>
</tbody>
</table>

### Related scenarios

For use cases in relation with **tParAccelOutputBulk**, see the following scenarios:

- **Scenario: Inserting transformed data in MySQL database** on page 2537.
- **Scenario: Inserting data in bulk in MySQL database** on page 2544.
- **Scenario: Truncating and inserting file data into an Oracle database** on page 2752.
tParAccelOutputBulkExec

Improves performance when loading data in ParAccel database.

The tParAccelOutputBulk and tParAccelBulkExec are generally used together in a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in tParAccelOutputBulkExec.

The tParAccelOutputBulkExec performs an Insert action on the data.

**tParAccelOutputBulkExec Standard properties**

These properties are used to configure tParAccelOutputBulkExec running in the Standard Job framework.

The Standard tParAccelOutputBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property type</strong></td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address. Currently, only localhost, 127.0.0.1 or the exact IP address of the local machine is allowed for proper functioning. In other words, the database server must be installed on the same machine where the Studio is installed or where the Job using tParAccelOutputBulkExec is deployed.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of the DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Database name.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data.</td>
</tr>
</tbody>
</table>
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

<table>
<thead>
<tr>
<th>Table</th>
<th>Name of the table to be written. Note that only one table can be written at a time</th>
</tr>
</thead>
</table>
| Action on table | On the table defined, you can perform one of the following operations:  
None: No operation is carried out.  
Drop and create table: The table is removed and created again.  
Create table: The table does not exist and gets created.  
Create table if not exists: The table is created if it does not exist.  
Drop table if exists and create: The table is removed if already exists and created again.  
Clear table: The table content is deleted. |
| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.  
Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.  
When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.  
You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com). |
| Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
• View schema: choose this option to view the schema only.  
• Change to built-in property: choose this option to change the schema to Built-in for local changes.  
• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No.
upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Copy mode</th>
<th>Select the copy mode you want to use from either:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>Standard mode, without optimisation.</td>
</tr>
<tr>
<td>Parallel</td>
<td>Allows you to use several internal ParAccel APIs in order to optimise loading speed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Filename</th>
<th>Name of the file to be generated and loaded.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warning:</td>
<td>This file is generated on the machine specified by the URI in the Host field so it should be on the same machine as the database server.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>File Type</th>
<th>Select the file type from the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row separator</td>
<td>String (ex: \n on Unix) to distinguish rows.</td>
</tr>
<tr>
<td>Fields terminated by</td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td>Append</td>
<td>Select this check box to add the new rows at the end of the file.</td>
</tr>
<tr>
<td>Explicit IDs</td>
<td>The ID is already present in the file to be loaded or will be set by the database.</td>
</tr>
<tr>
<td>Remove Quotes</td>
<td>Select this check box to remove quotation marks from the file to be loaded.</td>
</tr>
<tr>
<td>Max. Errors</td>
<td>Type in the maximum number of errors before your Job stops.</td>
</tr>
<tr>
<td>Date Format</td>
<td>Type in the date format to be used.</td>
</tr>
<tr>
<td>Time/Timestamp Format</td>
<td>Enter the date and hour format to be used.</td>
</tr>
<tr>
<td>Additional COPY Options</td>
<td>Enter the specific, customized ParAccel option that you want to use.</td>
</tr>
<tr>
<td>Log file</td>
<td>Browse to or enter the access path to the log file in your directory.</td>
</tr>
<tr>
<td>Logging level</td>
<td>Select the information type you want to record in your log file.</td>
</tr>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Usage**

| Usage rule     | This component covers all possible SQL database queries. It allows you to carry out actions on a table or |

2905
on the data of a table in a ParAccel database. It enables you to create a reject flow, with a Row > Reject link filtering the data in error. For a usage example, see Scenario: Retrieving data in error with a Reject link on page 2529.

**Limitation**

The database server must be installed on the same machine where the Studio is installed or where the Job using tParAccelOutputBulkExec is deployed, so that the component functions properly.

**Related scenarios**

For related scenarios, see:

- Scenario: Inserting transformed data in MySQL database on page 2537 of tMysqlOutputBulk.
- Scenario: Inserting data in bulk in MySQL database on page 2544 of tMysqlOutputBulkExec.
- Scenario: Truncating and inserting file data into an Oracle database on page 2752 of tOracleBulkExec.
**tParAccelRollback**

Avoids to commit part of a transaction involuntarily. The tParAccelRollback cancels the transaction commit in the connected DB.

**tParAccelRollback Standard properties**

These properties are used to configure tParAccelRollback running in the Standard Job framework. The Standard tParAccelRollback component belongs to the Databases family. The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tParAccelConnection component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tParAccel* components, especially with the tParAccelConnection and tParAccelCommit components. |
| Dynamic settings | Click the [+]{param} button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |
Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenario

For tParAccelRollback related scenario, see tMysqlRollback on page 2546.
tParAccelRow

Acts on the actual DB structure or on the data (although without handling data), depending on the nature of the query and the database. The SQLBuilder tool helps you write easily your SQL statements.

The tParAccelRow is the specific component for this database query. It executes the SQL query stated onto the specified database. The row suffix means the component implements a flow in the job design although it doesn’t provide output.

**tParAccelRow Standard properties**

These properties are used to configure tParAccelRow running in the Standard Job framework.

The Standard tParAccelRow component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

Note:

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Schema</td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td><strong>Table Name</strong></td>
<td>Name of the table to be read.</td>
</tr>
<tr>
<td><strong>Query type</strong></td>
<td>Either Built-in or Repository.</td>
</tr>
</tbody>
</table>
**Built-in**: Fill in manually the query statement or build it graphically using SQLBuilder.

**Repository**: Select the relevant query stored in the Repository. The **Query** field gets accordingly filled in.

**Guess Query**
- Click the **Guess Query** button to generate the query which corresponds to your table schema in the **Query** field.

**Query**
- Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

**Die on error**
- This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

### Advanced settings

**Propagate QUERY’s recordset**
- Select this check box to insert the result of the query into a **COLUMN** of the current flow. Select this column from the **use column** list.

  **Note:**
  - This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of **Object** and this component is usually followed by tParseRecordSet.

**Use PreparedStatement**
- Select this check box if you want to query the database using a PreparedStatement. In the **Set PreparedStatement Parameter** table, define the parameters represented by “?” in the SQL instruction of the **Query** field in the **Basic Settings** tab.

  **Parameter Index**: Enter the parameter position in the SQL instruction.

  **Parameter Type**: Enter the parameter type.

  **Parameter Value**: Enter the parameter value.

  **Note:**
  - This option is very useful if you need to execute the same query several times. Performance levels are increased.

**Commit every**
- Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.

**tStat Catcher Statistics**
- Select this check box to collect log data at the component level.
### Global Variables

| Global Variables | QUERY: the query statement being processed. This is a Flow variable and it returns a string. ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

### Usage

| Usage rule | This component offers the flexibility benefit of the DB query and covers all possible SQL queries. |

| Dynamic settings | Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

### Limitation

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find |
Related scenarios

For a related scenario, see:

- **Scenario: Combining two flows for selective output** on page 2558
- **Procedure** on page 625.
- **Scenario: Removing and regenerating a MySQL table index** on page 2552.
**tParAccelSCD**

Addresses Slowly Changing Dimension needs, reading regularly a source of data and logging the changes into a dedicated SCD table

The tParAccelSCD reflects and tracks changes in a dedicated ParAccel SCD table.

**tParAccelSCD Standard properties**

These properties are used to configure tParAccelSCD running in the Standard Job framework.

The Standard tParAccelSCD component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the Repository file where properties are stored. The following fields are pre-filled in using fetched data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Connection type</strong></th>
<th>Select the relevant driver on the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Name of the DB schema.</td>
</tr>
</tbody>
</table>
| **Username and Password** | DB user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Table**           | Name of the table to be written. Note that only one table can be written at a time. |

**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

**SCD Editor**

The SCD editor helps to build and configure the data flow for slowly changing dimension outputs.

For more information, see *SCD management methodology* on page 2566.
### Use memory saving Mode
Select this check box to maximize system performance.

### Source keys include Null
Select this check box to allow the source key columns to have Null values.

**Warning:**
Special attention should be paid to the uniqueness of the source key(s) values when this option is selected.

### Die on error
This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

### Advanced settings

#### End date time details
Specify the time value of the SCD end date time setting in the format of HH:mm:ss. The default value for this field is 12:00:00.
This field appears only when SCD Type 2 is used and Fixed year value is selected for creating the SCD end date.

#### Debug mode
Select this check box to display each step during processing entries in a database.

#### tStatCatcher Statistics
Select this check box to collect log data at the component level.

### Global Variables

#### Global Variables
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE_UPDATED</strong></td>
<td>the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_INSERTED</strong></td>
<td>the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_REJECTED</strong></td>
<td>the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as Output component. It requires an Input component and Row main link as input.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Limitation</td>
<td>This component does not support using SCD type 0 together with other SCD types.</td>
</tr>
</tbody>
</table>

## Related scenario

For related scenarios, see tMysqlSCD on page 2563.
**tParseRecordSet**

Parses a recordset rather than individual records from a table.

The tParseRecordSet parses a set of records from a database table or DB query and possibly returns single records.

**tParseRecordSet Standard properties**

These properties are used to configure tParseRecordSet running in the Standard Job framework.

The Standard tParseRecordSet component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Prev. Comp. Column list</th>
<th>Set the column from the database that holds the recordset.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema</strong> and <strong>Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>The schema is created and stored locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
<td></td>
</tr>
<tr>
<td><strong>View schema</strong></td>
<td>choose this option to view the schema only.</td>
</tr>
<tr>
<td><strong>Change to built-in property</strong></td>
<td>choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td><strong>Update repository connection</strong></td>
<td>choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td><strong>Attribute table</strong></td>
<td>Set the position value of each column for single records from the recordset.</td>
</tr>
</tbody>
</table>
**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | NB_LINE: the number of rows processed. This is an After variable and it returns an integer.  
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

**Usage**

| Usage rule | This component is used as intermediary component. It can be used as start component but only input parameters are thus allowed. |

| Limitation | This component is mainly designed for a use with the SP component Recordset feature. |

**Related Scenario**

For an example of tParseRecordSet in use, see Scenario: Using PreparedStatement objects to query data on page 2553.
**tPigAggregate**

Adds one or more additional columns to the output of the grouped data to create data to be used by Pig.

The tPigAggregate groups the original data by column and adds one or more additional columns to the output of preceding grouped data.

**tPigAggregate Standard properties**

These properties are used to configure tPigAggregate running in the Standard Job framework.

The Standard tPigAggregate component belongs to the Big Data and the Processing families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click [Edit schema](#) to make changes to the schema. If the current schema is of the [Repository](#) type, three options are available:  
- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content](#) window. |
| **Built-in**: The schema will be created and stored locally for this component only. Related topic: Talend Studio User Guide. | |
| **Repository**: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job designs. Related topic: see Talend Studio User Guide. | |
| **Group by** | Click the plus button to add one or more columns to set tuples in the source data as group condition. |
| **Operations** | Click the plus button to add one or more columns to generate one or more additional output columns based on conditions: |
**Additional Output Column**: Select a column in the original data as output column.

**Function**: Functions for operation on input data.

**Input Column**: Select a column in the original data as input column.

### Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Increase parallelism</strong></td>
<td>Select this check box to set the number of reduce tasks for the MapReduce Jobs.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is commonly used as intermediate step together with input component and output component.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prerequisites</strong></td>
<td>The Hadoop distribution must be properly installed, so as to guarantee the interaction with <strong>Talend Studio</strong>. The following list presents MapR related information for example.</td>
</tr>
<tr>
<td></td>
<td>• Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under \MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: <a href="http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr">http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr</a>.</td>
</tr>
</tbody>
</table>
Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

• Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

| Limitation | Knowledge of Pig scripts is required. |

**Related scenario**

For a tPigAggregate related scenario, Aggregating values and sorting data on page 121 of tAggregateRow.
tPigCode

Extends the functionalities of a Talend Job through using Pig scripts.

The tPigCode enters personalized Pig code to integrate it in Talend program. You can execute this code only once.

**tPigCode Standard properties**

These properties are used to configure tPigCode running in the Standard Job framework.

The Standard tPigCode component belongs to the Big Data and the Processing families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td>Built-In</td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Repository</td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Scripts</td>
<td>Type in Pig scripts you want to execute depending on the task you need to perform. For further information about Pig functions syntax, see Apache's documentation about Pig UDF: <a href="http://pig.apache.org">http://pig.apache.org</a></td>
</tr>
</tbody>
</table>
Pig components output tuples and automatically set up an alias for each tuple. When you use a tuple in your Pig script, you have to enter the right alias.

The alias syntax is `Component_ID_rowID_Result`, for example, `tPigCode_1_row2_Result`.

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable escape</td>
<td>Select this check box so that you can simply write plain Pig code in the <em>Scripts</em> field without need to bear in mind the escape characters, otherwise required for proper Java code generation.</td>
</tr>
</tbody>
</table>

**Global Variables**

| Global Variables | `ERROR_MESSAGE`: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the *Die on error* check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press `Ctrl + Space` to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

**Usage**

| Usage rule | This component is commonly used as intermediate step together with input component and output component. A *tPigCode* component can execute only one Pig Latin statement, therefore, if you need to execute multiple statements, you have to use a corresponding number of *tPigCode* components to run them, one after another. If a particular .jar file is required to execute a statement, you need to register that library file via the *tPigLoad* component that starts the Pig process in question. |

**Prerequisites**

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or...
libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error:

```
no MapRClient in java.library.path.
```

- Set the `-Djava.library.path` argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

| Limitation | Knowledge of Pig scripts is required. |

**Scenario: Selecting a column of data from an input file and store it into a local file**

This scenario applies only to a Talend solution with Big Data.

This scenario describes a three-component Job that selects a column of data that matches filter condition defined in `tPigCode` and stores the result into a local file.

![Diagram](image)

**Setting up the Job**

**Procedure**

1. Drop the following components from the Palette to the design workspace: `tPigCode`, `tPigLoad`, `tPigStoreResult`.
2. Right-click `tPigLoad` to connect it to `tPigCode` using a Row > Pig Combine connection.
3. Right-click `tPigCode` to connect it to `tPigStoreResult` using a Row > Pig Combine connection.
Loading the data

Procedure

1. Double-click tPigLoad to open its Basic settings view.

   ![tPigLoad component](image)

   - Property Type: Built-In
   - Schema: Built-In
   - Configuration: Local
   - Load function: PigStorage
   - Input file URI: /home/CustomerList
   - Field separator: ;
   - Compression: None
   - Force to compress output data: Off

2. Click the three-dot button next to Edit schema to add columns for tPigLoad.

   ![Column settings](image)

   - Click the plus button to add Name, Country and Age and click OK to save the setting.

3. Click the plus button to add Name, Country and Age and click OK to save the setting.

4. Select Local from the Mode area.

5. Fill in the Input filename field with the full path to the input file.
   
   In this scenario, the input file is CustomerList which contains rows of names, country names and age.

6. Select PigStorage from the Load function list.

7. Leave rest of the settings as they are.

Configuring the tPigCode component

Procedure

1. Double-click tPigCode component to open its Basic settings view.

   ![tPigCode component](image)
2. Click **Sync columns** to retrieve the schema structure from the preceding component.
3. Fill in the **Script Code** field with following expression:

   ```
   tPigCode_1_row2_RESULT = foreach tPigLoad_1_row1_RESULT generate $0 as name;
   ```

   This filter expression selects column *Name* from *CustomerList*.

**Saving the result data to a local file**

**Procedure**

1. Double-click **tPigStoreResult** to open its **Basic settings** view.

   2. Click **Sync columns** to retrieve the schema structure from the preceding component.
   3. Fill in the **Result file** field with the full path to the result file.
      
      In this scenario, the result is saved in *Result* file.
   4. Select **Remove result directory if exists**.
   5. Select **PigStorage** from the **Store function** list.
   6. Leave rest of the settings as they are.

**Executing the Job**

Save your Job and press **F6** to run it.
The *Result* file is generated containing the selected column of data.

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
</tr>
<tr>
<td>Silvia</td>
</tr>
<tr>
<td>Romeo</td>
</tr>
<tr>
<td>Ahmad</td>
</tr>
<tr>
<td>Toyota</td>
</tr>
<tr>
<td>Manik</td>
</tr>
<tr>
<td>Natasha</td>
</tr>
<tr>
<td>Billy</td>
</tr>
<tr>
<td>Eminem</td>
</tr>
<tr>
<td>Bill</td>
</tr>
<tr>
<td>erricka</td>
</tr>
<tr>
<td>Huamei</td>
</tr>
<tr>
<td>Selena</td>
</tr>
<tr>
<td>Julio</td>
</tr>
<tr>
<td>Pantalion</td>
</tr>
<tr>
<td>Simao</td>
</tr>
<tr>
<td>Nancy</td>
</tr>
<tr>
<td>Gaddafi</td>
</tr>
<tr>
<td>Zidane</td>
</tr>
<tr>
<td>Didi</td>
</tr>
<tr>
<td>Juan</td>
</tr>
<tr>
<td>Bob</td>
</tr>
<tr>
<td>Mario</td>
</tr>
<tr>
<td>Ricky</td>
</tr>
</tbody>
</table>
tPigCoGroup

Performs the Pig COGROUP operation to group and aggregate data incoming from multiple Pig flows. The tPigCoGroup groups data from as many inputs as needed incoming from its preceding Pig components and aggregates the grouped data using some given function before sending the data to the next Pig component.

**tPigCoGroup Standard properties**

These properties are used to configure tPigCoGroup running in the Standard Job framework. The Standard tPigCoGroup component belongs to the Big Data and the Processing families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  • View schema: choose this option to view the schema only.  
  • Change to built-in property: choose this option to change the schema to Built-in for local changes.  
  • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td><strong>Group by</strong></td>
<td>Click the [+] button to add one or more columns of the input flows to this Group by table so as to set these columns as group condition.</td>
</tr>
<tr>
<td><strong>Output mapping</strong></td>
<td>This table is automatically filled with the output schema you have defined using the Schema field. Then</td>
</tr>
</tbody>
</table>
complete this table to configure how the grouped data is aggregated in the output flow:

**Function**: select the function you need to use to aggregate a given column.

**Source schema**: select the input flow from which you aggregate the data.

**Expression**: select the column to be aggregated and if needed, edit expressions

### Advanced settings

<table>
<thead>
<tr>
<th>Group optimization</th>
<th>Select the Pig algorithm depending on the situation of the input data and the loader you are using to optimize the COGROUP operation. For further information, see Apache’s documentation about Pig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use partitioner</td>
<td>Select this check box to call a Hadoop partitioner in order to partition records and return the reduce task or tasks that each record should go to. Note that this partitioner class must be registered in the Register jar table provided by the tPigLoad component that starts the current Pig process.</td>
</tr>
<tr>
<td>Increase parallelism</td>
<td>Select this check box to set the number of reduce tasks for the MapReduce Jobs.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

### Usage

| Usage rule | This component is commonly used as intermediate step together with input component and output component. |
Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

Limitation

Knowledge of Pig scripts is required.

Scenario: aggregating data from two relations using COGROUP

This scenario applies only to a Talend solution with Big Data.

In this scenario, a four-component Job is designed to aggregate two relations on top of a given Hadoop cluster.
The two relations used in this scenario consist of the following sample data:

1. Alice, turtle, 17
   Alice, goldfish, 17
   Alice, cat, 17
   Bob, dog, 18
   Bob, cat, 18
   John, dog, 19
   Mary, goldfish, 16
   Bill, dog, 20

   This relation is composed of three columns that read owner, pet and age (of the owners).

2. Cindy, Alice
   Mark, Alice
   Paul, Bob
   Paul, Jane
   John, Mary
   William, Bill

   This relation provides a list of students' names alongside their friends, of which some are pet owners displayed in the first relation. Therefore, the schema of this relation contains two columns: student and friend.

Before replicating this scenario, you need to write the sample data into the HDFS system of the Hadoop cluster to be used. To do this, you can use tHDFSOutput. For further information about this component, see tHDFSOutput on page 1538.

The data used in this scenario is inspired by the examples that Pig's documentation uses to explain the GROUP and the GOGROUP operators. For related information, please see Apache's documentation for Pig.

**Linking the components**

**Procedure**

1. In the **Integration** perspective of the Studio, create an empty Job from the **Job Designs** node in the **Repository** tree view.
   For further information about how to create a Job, see **Talend Studio User Guide**.
2. In the workspace, enter the name of the component to be used and select this component from the list that appears. In this scenario, the components are two **tPigLoad** components, a
**tPigCoGroup** component and a **tPigStoreResult** component. One of the two **tPigLoad** components is used as the main loading component to connect to the Hadoop cluster to be used.

3. Connect the main **tPigLoad** component to **tPigCoGroup** using the Row > Main link.
4. Do the same to connect the second **tPigLoad** component to **tPigCoGroup**. The Lookup label appears over this link.
5. Repeat the operation to connect **tPigCoGroup** to **tPigStoreResult**.

### Reading data into the Pig flow

**Reading the owner-pet sample data**

#### Procedure

1. Double-click the main **tPigLoad** component to open its Component view.

   ![Diagram of tPigLoad component](image)

<table>
<thead>
<tr>
<th>Basic settings</th>
<th>Advanced settings</th>
<th>Dynamic settings</th>
<th>View</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type: Built-In</td>
<td>Schema: Built-In</td>
<td>Configuration: Local</td>
<td>Distribution: HortonWorks</td>
<td>Load function: PigStorage</td>
</tr>
<tr>
<td>Configuration: Local</td>
<td>Distribution: HortonWorks</td>
<td>Version: Hortonworks Data Platform V2.1.0 (Baikal)</td>
<td>Load function: PigStorage</td>
<td></td>
</tr>
<tr>
<td>NameNode URL: &quot;hdfs//tal-qa113.talend.land8020/&quot;</td>
<td>Resource Manager: &quot;tal-qa114.talend.land8050&quot;</td>
<td>JobHistory Address: &quot;tal-qa114.talend.land8030&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use DataNode Hostname</td>
<td>Use Kerberos Authentication</td>
<td>User Name: &quot;hdfs&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use S3 Endpoint</td>
<td>Input File URL: &quot;/user/ychen/pigco/owner.csv&quot;</td>
<td>Field Separator: &quot;;&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compression</td>
<td>Force to Compress the Output Data</td>
<td>Die on SubJob Error</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Click the [...] button next to **Edit schema** to open the schema editor and click the [+ ] button three times to add three rows.
3. In the Column column, rename the new rows to **owner**, **pet** and **age**, respectively, and in the Type column of the **age** row, select **Integer**.
4. Click **OK** to validate these changes and accept the propagation prompted by the pop-up dialog box.
5. In the Mode area, select **Map/Reduce** to use the remote Hadoop cluster to be used.
6. In the Distribution and the Version lists, select the Hadoop distribution you are using. In this example, *HortonWorks Data Platform V2.1.0 (Baikal)* is selected.
7. In the Load function list, select **PigStorage**. Then, the corresponding parameters to be set appear.
8. In the **NameNode URI** and the **Resource manager** fields, enter the locations of those services, respectively. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS.

9. Select the **Set Resourcemanager scheduler address** check box and enter the URI of this service in the field that is displayed. This allows you to use the Scheduler service defined in the Hadoop cluster to be used. If this service is not defined in your cluster, you can ignore this step.

10. In the **User name** field, enter the name of the user having the appropriate right to write data in the cluster. In this example, it is `hdfs`.

11. In the **Input file URI** field, enter the path pointing to the relation you need to read data from. As explained previously, the relation to be read here is the one containing the owner and pet sample data.

12. In the **Field separator** field, enter the separator of the data to be read. In this example, it is `;`.

### Loading the student-friend sample data

**Procedure**

1. Double-click the second **tPigLoad** component to open its **Component** view.

   ![Component view of tPigLoad](image)

   - **Basic settings**
     - **Property Type**: Built-In
     - **Schema**: Built-In
   - **Advanced settings**
     - **Configuration**: Local
     - **Distribution**: HortonWorks
     - **Version**: Hortonworks Data Platform V2.1.0(Baikal)
     - **Load function**: PigStorage
   - **Dynamic settings**
     - **Use S3 endpoint**: False
     - **Input file URI**: `/user/ychen/pigco/friends.csv`
   - **Field separator**: `;`
   - **Compression**: False
   - **Force to compress the output data**: False
   - **Die on subjob error**: False

2. Click the `[...]` button next to **Edit schema** to open the schema editor.

3. Click the `[+]` button twice to add two rows and in the **Column** column, rename them to `student` and `friend`, respectively.

---

2934
4. Click OK to validate these changes and accept the propagation prompted by the pop-up dialog box.

5. In the Mode area, select Map/Reduce.
   This component reuses the Hadoop connection you have configured in that main tPigLoad component. Therefore, the Distribution and the Version fields have been automatically filled with the values from that main loading component.

6. In the Load function field, select the PigStorage function to read the source data.

7. In the Input file URI field, enter the directory where the source data is stored. As explained previously, this data is from the second relation containing the student and friend sample data.

Aggregating the relations

Procedure

1. Double-click tPigCoGroup to open its Component view.

2. Click the [...] button next to Edit schema to open the schema editor.
3. Click the [+] button five times to add five rows and in the **Column** column, rename them to *owner_friend*, *age*, *pet_number*, *pet* and *student*, respectively.

4. In the **Type** column of the *age* row, select **Integer**.

5. Click **OK** to validate these changes and accept the propagation prompted by the pop-up dialog box.

6. In the **Group by** table, click the [+] button once to add one row.

7. Then you need to set the grouping condition in this **Group by** table to aggregate the two input relations. In each column representing the input relation, click the newly added row and select the column you need to use to compose the grouping condition. In this scenario, the *owner* column from the owner-pet relation and the *friend* column from the student-friend relation are selected because they have common records. Based on these columns, the two relations are aggregated into bags.

   The bags regarding the record *Alice* might read as follow:

   ```
   Alice,\{(Alice,turtle,17),(Alice,goldfish,17),(Alice,cat,17)\},\{(Cindy,Alice),(Mark,Alice)\}
   ```

8. In the **Output mapping** table, the output schema you defined previously has been automatically fed into the **Column** column. You need to complete this table to define how the grouped bags are aggregated into the schema of the output relation. The following list provides more details about how this aggregation is configured for this scenario:

   **Column** | **Description**
   --- | ---
   **owner_friend** | Receive the literal records incoming from the columns that are used as the grouping condition.
   
   For this reason, select the **EMPTY** function from the **Function** drop-down list so that the incoming records stay as is. Then select *row1* from the **Source schema** list and *owner* from the **Expression** list to read the records from the corresponding input column; you can as well select *row2* and *friend*, the records to be received are the same because the *owner* column and the *friend* column are joined when they are used as grouping condition.
   
   Note that the label *row1* is the ID of the input link and thus may be different in your scenario.

   **age** | Receive the age data.
   
   As shown in the example bags in the previous step, the age of an owner repetitively appears in one of the bags after the grouping. You can select the **AVG** function from the **Function** list to make the average of the repetitive values such that this age appears only once in the final result. Then select *row1* from the **Source schema** list and *age* from the **Expression** list.
<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pet_number</td>
<td>Receive how many pets an owner has.</td>
</tr>
<tr>
<td></td>
<td>Select the <strong>COUNT</strong> function from the <strong>Function</strong> list to perform this</td>
</tr>
<tr>
<td></td>
<td>calculation. Then select <strong>row1</strong> from the <strong>Source schema</strong> list and <strong>pet</strong></td>
</tr>
<tr>
<td></td>
<td>from the <strong>Expression</strong> list.</td>
</tr>
</tbody>
</table>

**pet and student**

Receive the grouped records from the input **pet** and **student** columns, respectively.

Select **EMPTY** for both of them and from the **Source schema** list of each, select the corresponding input schema and from the **Expression** list, the corresponding column.

**Writing the aggregated data**

**Procedure**

1. Double-click **tPigStoreResult** to open its **Component** view.

2. If this component does not have the same schema of the preceding component, a warning icon appears. In this case, click the **Sync columns** button to retrieve the schema from the preceding one and once done, the warning icon disappears.

3. In the **Result folder URI** field, enter the path in HDFS pointing to the location you want to write the result in.

4. Select the **Remove result directory if exists** check box.

5. From the **Store function** list, select **PigStorage**.

6. In the **Field separator** field, enter the separator you want to use. In this scenario, enter a comma (,).

**Executing the Job**

Then you can press **F6** to run this Job.

Once done, check the result from the HDFS system you are using.
You can read, for example, that the pet owner *Alice* is 17 years old, has 3 pets, a cat, a goldfish and a turtle and two of her friends are Mark and Cindy.
tPigCross

Uses CROSS operator to compute the Cartesian product.
The tPigCross computes the cross data of two or more relations.

tPigCross Standard properties

These properties are used to configure tPigCross running in the Standard Job framework.
The Standard tPigCross component belongs to the Big Data and the Processing families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
| • View schema: choose this option to view the schema only.  
| • Change to built-in property: choose this option to change the schema to Built-in for local changes.  
| • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
| Built-in: The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide. |
| Repository: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job designs. Related topic: see Talend Studio User Guide. |
| Cross filename | Fill in the path of the file to be computed. |
| Field separator | Enter character, string or regular expression to separate fields for the transferred data. |
## Advanced settings

<table>
<thead>
<tr>
<th>Use partitioner</th>
<th>Select this check box to specify the Hadoop Partitioner that controls the partitioning of the keys of the intermediate map-outputs. For further information about the usage of Hadoop Partitioner, see: <a href="http://hadoop.apache.org/docs/r2.2.0/api/org/apache/hadoop/mapred/Partitioner.html">http://hadoop.apache.org/docs/r2.2.0/api/org/apache/hadoop/mapred/Partitioner.html</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase parallelism</td>
<td>Select this check box to set the number of reduce tasks for the MapReduce Jobs</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

## Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it. For further information about variables, see <a href="#">Talend Studio User Guide</a>.</th>
</tr>
</thead>
</table>

## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is commonly used as intermediate step together with input component and output component.</th>
</tr>
</thead>
</table>
| Prerequisites | The Hadoop distribution must be properly installed, so as to guarantee the interaction with **Talend Studio**. The following list presents MapR related information for example.  

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr). |
Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the `-Djava.library.path` argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

| Limitation                  | Knowledge of Pig scripts is required. |

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tPigDistinct

Removes duplicate tuples in a relation.

**tPigDistinct Standard properties**

These properties are used to configure tPigDistinct running in the Standard Job framework.

The Standard tPigDistinct component belongs to the Big Data and the Processing families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in**: The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job designs. Related topic: see Talend Studio User Guide.

**Advanced settings**

Increase parallelism

Select this check box to set the number of reduce tasks for the MapReduce Jobs

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.
### Global Variables

| **Global Variables** | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

### Usage

| **Usage rule** | **This component is commonly used as intermediate step together with input component and output component.** **Warning:** This component will not maintain the original order in the input file. |

### Prerequisites

| **Prerequisites** | **The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.** **•** Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr. Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path. **•** Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR. |
For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

| Limitation          | Knowledge of Pig scripts is required. |

**Related scenario**

For more information regarding the **tPigDistinct** component in use, see Scenario: Filtering rows of data based on a condition and saving the result to a local file on page 2950 of **tPigFilterRow**.
**tPigFilterColumns**

Selects data or filters out data from a relation based on defined filter conditions.

The tPigFilterColumns selects one or more columns from a relation based on defined condition.

**tPigFilterColumns Standard properties**

These properties are used to configure tPigFilterColumns running in the Standard Job framework.

The Standard tPigFilterColumns component belongs to the Big Data and the Processing families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
</tbody>
</table>

| Built-in: The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide. |

| Repository: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job designs. Related topic: see Talend Studio User Guide. |

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>
### Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

### Usage

| Usage rule | This component is commonly used as intermediate step together with input component and output component. |

### Prerequisites

| Prerequisites | The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example. |

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr). Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`. |

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the **[Preferences]** dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the **Data viewer** to view locally in the Studio the data stored in MapR. |

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using. |

### Limitation

| Limitation | Knowledge of Pig script is required. |
Related Scenario

For a \texttt{tPigFilterColumns} related scenario, see Scenario: Joining two files based on an exact match and saving the result to a local file on page 2957 of \texttt{tPigJoin}. 
tPigFilterRow

Applies filtering conditions on one or more specified columns, in a Pig process, in order to split or filter data from a relation.

The tPigFilterRow filters or splits the input flow in a Pig process based on conditions set on given column(s).

**tPigFilterRow Standard properties**

These properties are used to configure tPigFilterRow running in the Standard Job framework.

The Standard tPigFilterRow component belongs to the Big Data and the Processing families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:
| | - View schema: choose this option to view the schema only.
| | - Change to built-in property: choose this option to change the schema to Built-in for local changes.
| | - Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.
| Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
| Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.
| Filter configuration | Click the Add button beneath the Filter configuration table to set one or more filter conditions.
| | Note that when the column to be used by a condition is of the string type, the text to be entered in the Value column must be surrounded by both single and double quotation marks (for example, "California"), because the
double quotation marks are required by Talend's code generator and the single quotation marks required by Pig's grammar.

**Note:**
This table disappears if you select **Use advanced filter**.

<table>
<thead>
<tr>
<th>Use advanced filter</th>
<th>Select this check box to define advanced filter condition by entering customized filter expression in the Filter field.</th>
</tr>
</thead>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

### Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |
|---------------------|------------------------------------------------------------------------------------------------------------------|

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is commonly used as an intermediate step in a Pig process.</th>
</tr>
</thead>
</table>

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR's documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/](http://www.mapr.com/)
Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the `-Djava.library.path` argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

| Limitation               | Knowledge of Pig scripts is required. |

**Scenario: Filtering rows of data based on a condition and saving the result to a local file**

This scenario applies only to a Talend solution with Big Data.

This scenario describes a four-component Job that filters a list of customers to find out customers from a particular country, and saves the result list to a local file. Before the input data is filtered, duplicate entries are first removed from the list.

The input file contains three columns: Name, Country, and Age, and it has some duplicate entries, as shown below:

<table>
<thead>
<tr>
<th>Name</th>
<th>Country</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mario</td>
<td>PuertoRico</td>
<td>49</td>
</tr>
<tr>
<td>Mike</td>
<td>USA</td>
<td>22</td>
</tr>
<tr>
<td>Ricky</td>
<td>PuertoRico</td>
<td>37</td>
</tr>
<tr>
<td>Silvia</td>
<td>Spain</td>
<td>20</td>
</tr>
<tr>
<td>Billy</td>
<td>Canada</td>
<td>21</td>
</tr>
<tr>
<td>Ricky</td>
<td>PuertoRico</td>
<td>37</td>
</tr>
<tr>
<td>Romeo</td>
<td>UK</td>
<td>19</td>
</tr>
<tr>
<td>Natasha</td>
<td>Russia</td>
<td>25</td>
</tr>
<tr>
<td>Juan</td>
<td>Cuba</td>
<td>23</td>
</tr>
<tr>
<td>Bob</td>
<td>Jamaica</td>
<td>55</td>
</tr>
<tr>
<td>Mario</td>
<td>PuertoRico</td>
<td>49</td>
</tr>
</tbody>
</table>

**Dropping and linking components**

**Procedure**

1. Drop the following components from the Palette to the design workspace: **tPigLoad**, **tPigDistinct**, **tPigFilterRow**, and **tPigStoreResult**.
2. Right-click **tPigLoad**, select Row > Pig Combine from the contextual menu, and click **tPigDistinct** to link these two components.
3. Repeat this operation to link **tPigDistinct** to **tPigFilterRow**, and **tPigFilterRow** to **tPigStoreResult** using Row > Pig Combine connections to form a Pig process.
Configuring the components

Loading the input data and removing duplicates

Procedure

1. Double-click **tPigLoad** to open its **Basic settings** view.

2. Click the [...] button next to **Edit schema** to open the [Schema] dialog box.
3. Click the \([+]\) button to add three columns according to the data structure of the input file: \textit{Name} (string), \textit{Country} (string) and \textit{Age} (integer), and then click \textbf{OK} to save the setting and close the dialog box.

4. Click \textbf{Local} in the \textit{Mode} area.

5. Fill in the \textit{Input file URI} field with the full path to the input file.

6. Select \textbf{PigStorage} from the \textit{Load function} list, and leave rest of the settings as they are.

7. Double-click \textbf{tPigDistinct} to open its \textbf{Basic settings} view, and click \textbf{Sync columns} to make sure that the input schema structure is correctly propagated from the preceding component.

   This component will remove any duplicates from the data flow.

### Configuring the filter

**Procedure**

1. Double-click \textbf{tPigFilterRow} to open its \textbf{Basic settings} view.

2. Click \textbf{Sync columns} to make sure that the input schema structure is correctly propagated from the preceding component.

3. Select \textbf{Use advanced filter} and fill in the \textbf{Filter} field with filter expression:

   \[
   "\text{Country matches 'PuertoRico'}"
   \]

   This filter expression selects rows of data that contains "PuertoRico" in the \textit{Country} column.
Configuring the file output

Procedure

1. Double-click tPigStoreResult to open its Basic settings view.

![tPigStoreResult Settings](image)

2. Click Sync columns to make sure that the input schema structure is correctly propagated from the preceding component.

3. Fill in the Result file field with the full path to the result file.

4. If the target file already exists, select the Remove result directory if exists check box.

5. Select PigStorage from the Store function list, and leave rest of the settings as they are.

Saving and executing the Job

Procedure

1. Press Ctrl+S to save your Job.

2. Press F6 or click the Run button on the Run tab to run the Job.

The result file contains the information of customers from the specified country.

![Result File](image)
tPigJoin

Performs inner joins and outer joins of two files based on join keys to create data that will be used by Pig.
The tPigJoin performs join of two files based on join keys.

tPigJoin Standard properties

These properties are used to configure tPigJoin running in the Standard Job framework.
The Standard tPigJoin component belongs to the Big Data and the Processing families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td>To make this component work, two schemas must be set: the schema of the main flow and the schema of the lookup flow. In the output part of the main schema, the columns of the main input file must be manually concatenated with those of the lookup file.</td>
</tr>
<tr>
<td>Built-in: The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Repository: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>
**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.

**Note:**

To make this component work, two schemas must be set: the schema of the main flow and the schema of the lookup flow. In the output part of the main schema, the columns of the main input file must be manually concatenated with those of the lookup file.

| **Built-in**: The schema will be created and stored locally for this component only. Related topic: see [Talend Studio User Guide](#). |
| **Repository**: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job designs. Related topic: see [Talend Studio User Guide](#). |

**Filename**

Fill in the path of the **Lookup file**.

**Field Separator**

Enter character, string or regular expression to separate fields for the transferred data.

**Join key**

Click the plus button to add lines to set the **Join key** for **Input file** and **Lookup file**.

**Join mode**

Select a join mode from the list:

- **inner-join**: Select this mode to perform an inner join of two or more relations based on Join keys.
- **left-outer-join**: Select this mode to perform a left outer join of two or more relations based on Join keys.
- **right-outer-join**: Select this mode to perform a right outer join of two or more relations based on Join keys.
- **full-outer-join**: Select this mode to combine the effect of applying both left and right outer joins.
For further information about inner join and outer join, see:
http://en.wikipedia.org/wiki/Join_%28SQL%29

**Advanced settings**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimize the join</td>
<td>Select this check box to optimize the performance of joins using <strong>REPLICATED</strong>, <strong>SKEWED</strong>, or <strong>MERGE</strong> joins. For further information about optimized joins, see: <a href="http://pig.apache.org/docs/r0.8.1/piglatin_ref1.html#Specialized+Joins">http://pig.apache.org/docs/r0.8.1/piglatin_ref1.html#Specialized+Joins</a></td>
</tr>
<tr>
<td>Use partitioner</td>
<td>Select this check box to specify the Hadoop Partitioner that controls the partitioning of the keys of the intermediate map-outputs. For further information about the usage of Hadoop Partitioner, see: <a href="http://hadoop.apache.org/docs/r2.2.0/api/org/apache/hadoop/mapred/Partitioner.html">http://hadoop.apache.org/docs/r2.2.0/api/org/apache/hadoop/mapred/Partitioner.html</a></td>
</tr>
<tr>
<td>Increase parallelism</td>
<td>Select this check box to set the number of reduce tasks for the <strong>MapReduce</strong> Jobs</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

**Usage rule**

This component is commonly used as intermediate step together with input component and output component.

**Prerequisites**

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.
**Scenario: Joining two files based on an exact match and saving the result to a local file**

This scenario applies only to a Talend solution with Big Data.

This scenario describes a four-component Job that combines data of an input file and a reference file that matches a given join key, removes unwanted columns, and then saves the final result to a local file.

The main input file contains the information about people’s IDs, first names, last names, group IDs, and salaries, as shown below:

```
1;Woodrow;Johnson;3;1013.39
2;Millard;Monroe;2;8077.59
3;Calvin;Eisenhower;3;6866.88
4;Lyndon;Wilson;3;5726.28
5;Ronald;Garfield;2;4158.58
6;Rutherford;Buchanan;3;2897.00
7;Calvin;Coolidge;1;6650.66
8;Ulysses;Roosevelt;2;7854.78
9;Grover;Tyler;1;5226.88
10;Bill;Tyler;2;8964.66
```

The reference file contains only the information of group IDs and group names:

```
1;group_A
```
Dropping and linking the components

Procedure

1. Drop the following components from the Palette to the design workspace: tPigLoad, tPigJoin, tPigFilterColumns, and tPigStoreResult.
2. Connect these components in a series using Row > Pig Combine connections.

Configuring the components

Loading the main input file

Procedure

1. Double-click tPigLoad to open its Basic settings view.
2. Click the [...] button next to Edit schema to open the [Schema] dialog box.
3. Click the [+] button to add columns, name them and define the column types according to the structure of the input file. In this example, the input schema has five columns: id (integer), firstName (string), lastName (string), groupid (integer), and salary (double). Then click OK to validate the setting and close the dialog box.

4. Click Local in the Mode area.

5. Select PigStorage from the Load function list.

6. Fill in the Input file URI field with the full path to the input file, and leave the rest of the setting as they are.

**Loading the reference file and setting up an inner join**

**Procedure**

1. Double-click tPigJoin to open its Basic settings view.
2. Click the [...] for the main schema to open the [Schema] dialog box.

3. Check that input schema is correctly retrieved from the preceding component. If needed, click the [...] button to copy all the columns of the input schema to the output schema.

4. Click the [+] button under the output panel to add new columns according to the data structure of the reference file, groupId_ref (integer) and groupName (string) in this example. Then click OK to close the dialog box.

5. Click the [...] for the schema lookup flow to open the [Schema] dialog box.
6. Click the [+] button under the output panel to add two columns: \textit{groupId\_ref} (integer) and \textit{groupName} (string), and then click \textbf{OK} to close the dialog box.

7. In the \textbf{Filename} field, specify the full path to the reference file.

8. Click the [+] button under the \textbf{Join key} table to add a new line, and select \textit{groupId} and \textit{groupId\_ref} respectively from the \textbf{Input} and \textbf{Lookup} lists to match data from the main input flow with data from the lookup flow based on the group ID.

9. From the \textbf{Join Mode} list, select \textbf{inner-join}.

**Defining the final output schema and the output file**

**Procedure**

1. Double-click \textbf{tPigFilterColumns} to open its \textbf{Basic settings} view.

2. Click the [...] button next to \textbf{Edit schema} to open the [Schema] dialog box.
3. From the input schema, select the columns you want to include in your result file by clicking them one after another while pressing the Shift key, and click the [->] button to copy them to the output schema. Then, click OK to validate the schema setting and close the dialog box. In this example, we want the result file to include all the information except the group IDs.

4. Double-click tPigStoreResult to open its Basic settings view.

5. Click Sync columns to retrieve the schema structure from the preceding component.

6. Fill in the Result file field with the full path to the result file, and select the Remove result file directory if exists check box.

7. Select PigStorage from the Store function list, and leave rest of the settings as they are.

Saving and executing the Job

Procedure

1. Press Ctrl+S to save your Job.

2. Press F6 or click Run on the Run tab to run the Job.

   The result file includes all the information related to people of group A and group B, except their group IDs.
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7;Calvin;Coolidge;6650.66;group_A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>9;Grover;Tyler;5226.88;group_A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2;Millard;Monroe;8077.39;group_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8;Ronald;Garfield;4158.58;group_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0;Ulysses;Roosevelt;7834.78;group_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10;Bill;Tyler;8964.66;group_B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**tPigLoad**

Loads original input data to an output stream in just one single transaction, once the data has been validated.

The tPigLoad sets up a connection to the data source for a current transaction.

**tPigLoad Standard properties**

These properties are used to configure tPigLoad running in the Standard Job framework.

The Standard tPigLoad component belongs to the Big Data and the Processing families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file where the properties are stored.</td>
</tr>
<tr>
<td></td>
<td>The properties are stored centrally under the Hadoop Cluster node of the Repository tree.</td>
</tr>
<tr>
<td></td>
<td>The fields that come after are pre-filled in using the fetched data.</td>
</tr>
<tr>
<td></td>
<td>For further information about the Hadoop Cluster node, see the Getting Started Guide.</td>
</tr>
</tbody>
</table>

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.
<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Local</strong></td>
<td>Click this radio button to run Pig scripts in Local mode. In this mode, all files are installed and run from your local host and file system.</td>
</tr>
</tbody>
</table>
| **Tez** | Click this radio button to run the Pig Job on the Tez framework. This Tez mode is available only when you are using one of the following distributions:  
- Hortonworks: V2.2 +.  
- Custom: this option allows you connect to a distribution supporting Tez but not officially supported by Talend.  
Before using Tez, ensure that the Hadoop cluster you are using supports Tez. You will need to configure the access to the relevant Tez libraries via the Advanced settings view of this component.  
For further information about Pig on Tez, see Apache’s related documentation in https://cwiki.apache.org/confluence/display/PIG/Pig+on+Tez. |
| **Map/Reduce** | Click this radio button to run Pig scripts in Map/Reduce mode. Once selecting this mode, you need to complete the fields in the Configuration area that appears:  
- **Distribution** and **Version**: Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:  
  - If available in this **Distribution** drop-down list, the Microsoft HD Insight option allows you use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).  
  - If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com). |
• The **Custom** option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by **Talend**.

1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In **Talend Exchange**, members of **Talend** community have shared some ready-for-use configuration zip files which you can download from this [Hadoop configuration list](#) and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**

In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see [Connecting to a custom Hadoop distribution](#) on page 1589.

• **Use Kerberos authentication:**

  If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.
• If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

In addition, since this component performs Map/Reduce computations, you also need to authenticate the related services such as the Job history server and the Resource manager or Jobtracker depending on your distribution in the corresponding field. These principals can be found in the configuration files of your distribution. For example, in a CDH4 distribution, the Resource manager principal is set in the `yarn-site.xml` file and the Job history principal in the `mapred-site.xml` file.

This check box is available depending on the Hadoop distribution you are connecting to.

The HBase related principals are required by the **HBaseStorage** function only.

• **Use a keytab to authenticate:**

  Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

  Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used.

• **NameNode URI:**

  Type in the location of the NameNode corresponding to the Map/Reduce version to be used. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS.
• **JobTracker host:**

  Type in the location of the ResourceManager corresponding to the Map/Reduce version to be used.

  In JobHistory, you can easily find the execution status of your Pig Job because the name of the Job is automatically created by concatenating the name of the project that contains the Job, the name and version of the Job itself and the label of the first `tPigLoad` component used in it. The naming convention of a Pig Job in JobHistory is `ProjectName_JobNameVersion_FirstComponentName`.

  Then you can continue to set the following parameters depending on the configuration of the Hadoop cluster to be used (if you leave the check box of a parameter clear, then at runtime, the configuration about this parameter in the Hadoop cluster to be used will be ignored):

  1. Select the **Set resourcemanager scheduler address** check box and enter the Scheduler address in the field that appears.

  2. Select the **Set jobhistory address** check box and enter the location of the JobHistory server of the Hadoop cluster to be used. This allows the metrics information of the current Job to be stored in that JobHistory server.

  3. Select the **Set staging directory** check box and enter this directory defined in your Hadoop cluster for temporary files created by running programs. Typically, this directory can be found under the `yarn.app.mapreduce.am.staging-dir` property in the configuration files such as `yarn-site.xml` or `mapred-site.xml` of your distribution.

  4. Allocate proper memory volumes to the **Map** and the **Reduce** computations and the **ApplicationMaster** of YARN by selecting the **Set memory** check box in the Advanced settings view.

  5. Select the **Set Hadoop user** check box and enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed.

  6. Select the **Use datanode hostname** check box to allow the Job to access datanodes via their hostnames. This actually sets the `dfs.client.use.datanode.hostname` property to `true`. When connecting to a S3N filesystem, you must select this check box.

For further information about these parameters, see the documentation or contact the administrator of the Hadoop cluster to be used.
**User name:**

Enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed. Note that this field is available depending on the distribution you are using.

**WebHCat configuration**

Enter the address and the authentication information of the Microsoft HD Insight cluster to be used. For example, the address could be `your_hdinsight_cluster_name.azurehdinsight.net` and the authentication information is your Azure account name: `ychen`. The Studio uses this service to submit the Job to the HD Insight cluster.

In the **Job result folder** field, enter the location in which you want to store the execution result of a Job in the Azure Storage to be used.

**HDInsight configuration**

Enter the authentication information of the HD Insight cluster to be used.

**Windows Azure Storage configuration**

Enter the address and the authentication information of the Azure Storage account to be used. In this configuration, you do not define where to read or write your business data but define where to deploy your Job only. Therefore always use the Azure Blob Storage system for this configuration.

In the **Container** field, enter the name of the container to be used.

In the **Deployment Blob** field, enter the location in which you want to store the current Job and its dependent libraries in this Azure Storage account.

**Inspect the classpath for configurations**

Select this check box to allow the component to check the configuration files in the directory you have set with the `$HADOOP_CONF_DIR` variable and directly read parameters from these files in this directory. This feature allows you to easily change the Hadoop configuration for the component to switch between different environments, for example, from a test environment to a production environment.

In this situation, the fields or options used to configure Hadoop connection and/or Kerberos security are hidden.

If you want to use certain parameters such as the Kerberos parameters but these parameters are not included in these Hadoop configuration files, you need to create a file called `talend-site.xml` and put this file into the same directory defined with `$HADOOP_CONF_DIR`. This `talend-site.xml` file should read as follows:

```xml
<!-- Put site-specific property overrides in this file. -->
<configuration>
    <property>
```

2969

```xml
```
<name>talend.kerberos.authentication</name>
  <value>kinit</value>
  <description> Set the Kerberos authentication method to use. Valid values are: kinit or keytab. </description>
</property>

<property>
  <name>talend.kerberos.keytab.principal</name>
  <value>user@BIGDATA.COM</value>
  <description> Set the keytab's principal name. </description>
</property>

<property>
  <name>talend.kerberos.keytab.path</name>
  <value>/kdc/user.keytab</value>
  <description> Set the keytab's path. </description>
</property>

<property>
  <name>talend.encryption</name>
  <value>none</value>
  <description> Set the encryption method to use. Valid values are: none or ssl. </description>
</property>

<property>
  <name>talend.ssl.trustStore.path</name>
  <value>ssl</value>
  <description> Set SSL trust store path. </description>
</property>

<property>
  <name>talend.ssl.trustStore.password</name>
  <value>ssl</value>
  <description> Set SSL trust store password. </description>
</property>

The parameters read from these configuration files override the default ones used by the Studio. When a parameter does not exist in these configuration files, the default one is used.

<table>
<thead>
<tr>
<th><strong>Load function</strong></th>
<th>Select a load function for data to be loaded:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• <strong>PigStorage</strong>: Loads data in UTF-8 format.</td>
<td></td>
</tr>
<tr>
<td>• <strong>BinStorage</strong>: Loads data in machine-readable format.</td>
<td></td>
</tr>
</tbody>
</table>
- **TextLoader**: Loads unstructured data in UTF-8 format.

- **HCatLoader**: Loads data from HCataLog managed tables using Pig scripts.
  
  This function is available only when you have selected HortonWorks as the Hadoop distribution to be used from the Distribution and the Version fields displayed in the Map/Reduce mode. For further information about HCatLoader, see [http://hive.apache.org/javadocs/hcat-r0.5.0/api/org/apache/hcatalog/pig/HCatLoader.html](http://hive.apache.org/javadocs/hcat-r0.5.0/api/org/apache/hcatalog/pig/HCatLoader.html).

- **HBaseStorage**: Loads data from HBase. Then you need to complete the HBase configuration in the HBase configuration area displayed.

- **SequenceFileLoader**: Loads data of the SequenceFile formats. Then you need to complete the configuration of the file to be loaded in the Sequence Loader Configuration area that appears. This function is for the Map/Reduce mode only.

- **RCFilePigStorage**: Loads data of the RCFile format. This function is for the Map/Reduce mode only.

- **AvroStorage**: Loads Avro files. For further information about AvroStorage, see Apache’s documentation on [https://cwiki.apache.org/confluence/display/PIG/AvroStorage](https://cwiki.apache.org/confluence/display/PIG/AvroStorage). This function is for the Map/Reduce mode only.

- **ParquetLoader**: Loads Parquet file. This function is for the Map/Reduce mode only.

- **Custom**: Loads data using any user-defined load function. To do this, you need to register, in the Advanced settings tab view, the jar file containing the function to be used, and then, in the field displayed next to this Load function field, specify that function.

  For example, after registering a jar file called piggybank.jar, you can enter `org.apache.pig.piggybank.storage.XMLLoader('attr') as (xml:chararray)` to use the custom function, XMLLoader contained in that jar file. For further information about this piggybank.jar file, see [https://cwiki.apache.org/confluence/display/PIG/PiggyBank](https://cwiki.apache.org/confluence/display/PIG/PiggyBank).

  Note that when the file format to be used is PARQUET, you might be prompted to find the specific Parquet jar file and install it into the Studio.

- When the connection mode to Hive is **Embedded**, the Job is run in your local machine and calls this jar installed in the Studio.

- When the connection mode to Hive is **Standalone**, the Job is run in the server hosting Hive and this jar file is sent to the HDFS system of the cluster you are connecting to. Therefore, ensure that you have properly defined the NameNode URI in the corresponding field of the Basic settings view.
This jar file can be downloaded from Apache's site. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

| Input file URI | Fill in this field with the full local path to the input file.  
| Note: This field is not available when you select HCatLoader from the Load function list or when you are using an S3 endpoint. |

| Use S3 endpoint | Select this check box to read data from a given Amazon S3 bucket folder.  
| Once this Use S3 endpoint check box is selected, you need to enter the following parameters in the fields that appear:  
| • S3 bucket name and folder: enter the bucket name and its folder from which you need to read data. You need to separate the bucket name and the folder name using a slash (/).  
| • Access key and Secret key: enter the authentication information required to connect to the Amazon S3 bucket to be used.  
| To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.  
| Note that the format of the S3 file is S3N (S3 Native Filesystem). |

| HCataLog Configuration | Fill the following fields to configure HCataLog managed tables on HDFS (Hadoop distributed file system):  
| Distribution and Version:  
| Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones require specific configuration:  
| • If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).  
| • If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).  
| • The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend. |
| **HCat metastore**: Enter the location of the HCatalog’s metastore, which is actually Hive’s metastore, a system catalog. For further information about Hive and HCatalog, see [http://hive.apache.org/](http://hive.apache.org/). |
| **Database**: The database in which tables are placed. |
| **Table**: The table in which data is stored. |
| **Partition filter**: Fill this field with the partition keys to list partitions by filter. |

Note: **HCataLog Configuration area** is enabled only when you select **HCatLoader** from the **Load function** list. For further information about the usage of HCatalog, see [https://cwiki.apache.org/confluence/display/Hive/HCatalog](https://cwiki.apache.org/confluence/display/Hive/HCatalog). For further information about the usage of **Partition filter**, see [https://cwiki.apache.org/confluence/display/HCATALOG/Design+Document+-+Java+APIs+for+HCatalog+DDL+Commands](https://cwiki.apache.org/confluence/display/HCATALOG/Design+Document+-+Java+APIs+for+HCatalog+DDL+Commands).

| **Field separator** | Enter character, string or regular expression to separate fields for the transferred data. |
| **Note:** | This field is enabled only when you select **PigStorage** from the **Load function** list. |

| **Compression** | Select the **Force to compress the output data** check box to compress the data when the data is outputted by **tPigStoreResult** at the end of a Pig process. |

Hadoop provides different compression formats that help reduce the space needed for storing files and speed up data transfer. When you need to write and compress data using the Pig program, by default you have to add a compression format as a suffix to the path pointing to the folder in which you want to write data, for example, `/user/ychen/out.bz2`. However, if you select this check box, the output data will be compressed even if you do not add any compression format to that path, such as `/user/ychen/out`. |

**Note:** The output path is set in the **Basic settings** view of **tPigStoreResult**. |

<p>| <strong>HBase configuration</strong> | This area is available to the HBaseStorage function. The parameters to be set are: |
| <strong>Zookeeper quorum</strong>: | Type in the name or the URL of the Zookeeper service you use to coordinate the transaction between your Studio and your database. Note that when you configure the Zookeeper, you might need to explicitly set the Zookeeper.znode.parent property to define the path to the root znode that contains all the znodes created and used by your database; then select the <strong>Set Zookeeper znode parent</strong> check box to define this property. |</p>
<table>
<thead>
<tr>
<th><strong>Zookeeper client port:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type in the number of the client listening port of the Zookeeper service you are using.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Table name:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter the name of the HBase table you need to load data from.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Load key:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Select this check box to load the row key as the first column of the result schema. In this situation, you must have created this column in the schema.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Mapping:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete this table to map the columns of the table to be used with the schema columns you have defined for the data flow to be processed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Sequence Loader configuration</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>This area is available only to the <strong>SequenceFileLoader</strong> function. Since a SequenceFile record consists of binary key/value pairs, the parameters to be set are:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Key column:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the Key column of a key/value record.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Value column</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the Value column of a key/value record.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Die on subjob error</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>This check box is cleared by default, meaning to skip the row on subjob error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Tez lib</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Select how the Tez libraries are accessed:</td>
</tr>
</tbody>
</table>

- **Auto install:** at runtime, the Job uploads and deploys the Tez libraries provided by the Studio into the directory you specified in the **Install folder in HDFS** field, for example, `/tmp/usr/tez`. |
  
  If you have set the `tez.lib.uris` property in the properties table, this directory overrides the value of that property at runtime. But the other properties set in the properties table are still effective. |

- **Use exist:** the Job accesses the Tez libraries already deployed in the Hadoop cluster to be used. You need to enter the path pointing to those libraries in the **Lib path (folder or file)** field. |

- **Lib jar:** this table appears when you have selected **Auto install** from the **Tez lib** list and the distribution you are using is **Custom**. In this table, you need to add the Tez libraries to be uploaded. |

<table>
<thead>
<tr>
<th><strong>Hadoop Properties</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Talend Studio uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or</td>
</tr>
</tbody>
</table>
Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).

- Apache also provides a page to list the Hive-related properties: [https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties](https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties).

### Register jar
Click the `[+]` button to add rows to the table and from these rows, browse to the jar files to be added. For example, in order to register a jar file called `piggybank.jar`, click the `[+]` button once to add one row, then click this row to display the `[...]` browse button, and click this button to browse to the `piggybank.jar` file following the [Select Module] wizard.

### Define functions
Use this table to define UDFs (User-Defined Functions), especially those requiring alias such as Apache DataFu Pig functions, to be executed when loading data.

Click the `[+]` button to add as many rows as you need and specify an alias and a UDF in the relevant fields for each row.

If your Job includes a **tPigMap** component, once you have defined UDFs for this component in the **tPigMap**, this table is automatically filled. Likewise, once you have defined UDFs in this table, the **Define functions** table in the **tPigMap** component’s Map Editor is automatically filled.

For information on how to define UDFs when mapping Pig flows, see the section on mapping Big Data flows of the *Talend Open Studio for Big Data Getting Started Guide*.

For more information on Apache DataFu Pig, see [http://datafu.incubator.apache.org/](http://datafu.incubator.apache.org/).

### Pig properties
*Talend Studio* uses a default configuration for its Pig engine to perform operations. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be
customized. Then at runtime, the customized property or properties will override those default ones.

For example, the `default_parallel` key used in Pig could be set as 20.

### HBaseStorage configuration

Add and set more HBaseStorage loader options in this table. The options are:

- **gt**: the minimum key value;
- **lt**: the maximum key value;
- **gte**: the minimum key value (included);
- **lte**: the maximum key value (included);
- **limit**: maximum number of rows to retrieve per region;
- **caching**: number of rows to cache;
- **caster**: the converter to use for reading values out of HBase. For example, HBaseBinaryConverter.

### Define the jars to register for HCatalog

This check box appears when you are using `tHCatLoader`, while you can leave it clear as the Studio registers the required jar files automatically. In case any jar file is missing, you can select this check box to display the `Register jar for HCatalog` table and set the correct path to that missing jar.

### Path separator in server

Leave the default value of the `Path separator in server` as it is, unless you have changed the separator used by your Hadoop distribution's host machine for its PATH variable or in other words, that separator is not a colon (:). In that situation, you must change this value to the one you are using in that host.

### Mapred job map memory mb and Mapred job reduce memory mb

You can tune the map and reduce computations by selecting the `Set memory` check box to set proper memory allocations for the computations to be performed by the Hadoop system.

In that situation, you need to enter the values you need in the `Mapred job map memory mb` and the `Mapred job reduce memory mb` fields, respectively. By default, the values are both 1000 which are normally appropriate for running the computations.

The memory parameters to be set are **Map (in Mb)**, **Reduce (inMb)** and **ApplicationMaster (in Mb)**. These fields allow you to dynamically allocate memory to the map and the reduce computations and the ApplicationMaster of YARN.

### tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an **After**
variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | This component is always used to start a Pig process and needs **tPigStoreResult** at the end to output its data. In the **Map/Reduce** mode, you need only configure the Hadoop connection for the first **tPigLoad** component of a Pig process (a subjob), and any other **tPigLoad** component used in this process reuses automatically that connection created by that first **tPigLoad** component. |

### Prerequisites

<table>
<thead>
<tr>
<th>Prerequisites</th>
<th>The Hadoop distribution must be properly installed, so as to guarantee the interaction with <em>Talend Studio</em>. The following list presents MapR related information for example.</th>
</tr>
</thead>
</table>
| • Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`. |
| • Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the **[Preferences]** dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the **Data viewer** to view locally in the Studio the data stored in MapR. |

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using. |
Limitation

Knowledge of Pig scripts is required. If you select HCatLoader as the load function, knowledge of HCatalog DDL (HCatalog Data Definition Language, a subset of Hive Data Definition Language) is required. For further information about HCatalog DDL, see https://cwiki.apache.org/confluence/display/Hive/HCatalog.

Scenario: Loading an HBase table

This scenario applies only to a Talend solution with Big Data.

This scenario uses **tPigLoad** and **tPigStoreResult** to read data from HBase and to write them to HDFS.

```
thbasestorage
```

The HBase table to be used has three columns: **id**, **name** and **age**, among which **id** and **age** belong to the column family, **family1** and **name** to the column family, **family2**.

The data stored in that HBase table are as follows:

```
1;Albert;23
2;Alexandre;24
3;Alfred-Hubert;22
4;Andre;40
5;Didier;28
6;Anthony;35
7;Artus;32
8;Catherine;34
9;Charles;21
10;Christophe;36
11;Christian;67
12;Danniel;54
13;Elisabeth;58
14;Emile;32
15;Gregory;30
```

To replicate this scenario, perform the following operations:

**Linking the components**

**Procedure**

1. In the **Integration** perspective of **Talend Studio**, create an empty Job, named **hbase_storage** for example, from the **Job Designs** node in the **Repository** tree view.
   
   For further information about how to create a Job, see the **Talend Studio User Guide**.

2. Drop **tPigLoad** and **tPigStoreResult** onto the workspace.

3. Connect them using the **Row > Pig combine** link.
Configuring tPigLoad

Procedure

1. Double-click **tPigLoad** to open its **Component** view.

2. Click the button next to **Edit schema** to open the schema editor.

3. Click the button four times to add four rows and rename them: *rowkey*, *id*, *name* and *age*. The *rowkey* column put at the top of the schema to store the HBase row key column, but in practice, if you do not need to load the row key column, you can create only the other three columns in your schema.
4. Click OK to validate these changes and accept the propagation prompted by the pop-up dialog box.
5. In the Mode area, select Map/Reduce, as we are using a remote Hadoop distribution.
6. In the Distribution and the Version fields, select the Hadoop distribution you are using. In this example, we are using HortonWorks Data Platform V1.
7. In the Load function field, select HBaseStorage. Then, the corresponding parameters to set appear.
8. In the NameNode URI and the Resource Manager fields, enter the locations of those services, respectively. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS.
9. In the Zookeeper quorum and the Zookeeper client port fields, enter the location information of the Zookeeper service to be used.
10. If the Zookeeper znode parent location has been defined in the Hadoop cluster you are connecting to, you need to select the Set zookeeper znode parent check box and enter the value of this property in the field that is displayed.
11. In the Table name field, enter the name of the table from which tPigLoad reads the data.
12. Select the Load key check box if you need to load the HBase row key column. In this example, we select it.
13. In the Mapping table, four rows have been added automatically. In the Column family:qualifier column, enter the HBase columns you need to map with the schema columns you defined. In this scenario, we put family1:id for the id column, family2:name for the name column and family1:age for the age column.

Configuring tPigStoreResult

Procedure
1. Double-click tPigStoreResult to open its Component view.
2. In the **Result file** field, enter the directory where you need to store the result. As **tPigStoreResult** reuses automatically the connection created by **tPigLoad**, the path in this scenario is the directory in the machine hosting the Hadoop distribution to be used.

3. Select **Remove result directory if exists**.

4. In the **Store function** field, select **PigStorage** to store the result in the UTF-8 format.

### Executing the Job

Then you can press **F6** run this Job.

Once done, you can verify the result in the HDFS system used.

If you need to obtain more details about the Job, it is recommended to use the web console of the Jobtracker provided by the Hadoop distribution you are using.

In JobHistory, you can easily find the execution status of your Pig Job because the name of the Job is automatically created by concatenating the name of the project that contains the Job, the name and version of the Job itself and the label of the first **tPigLoad** component used in it. The naming convention of a Pig Job in JobHistory is `ProjectName_JobNameVersion_FirstComponentName`. 
tPigMap

Transforms and routes data from single or multiple sources to single or multiple destinations.

The tPigMap transforms and routes the data in a Pig process. It provides a graphic interface that enables sophisticated configuration of multiple data flows.

Optional map settings

On the input side:

<table>
<thead>
<tr>
<th>Lookup properties</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Join Model</td>
<td>Inner Join;</td>
</tr>
<tr>
<td></td>
<td>Left Outer Join;</td>
</tr>
<tr>
<td></td>
<td>Right Outer Join;</td>
</tr>
<tr>
<td></td>
<td>Full Outer Join.</td>
</tr>
<tr>
<td></td>
<td>The default join option is <strong>Left Outer Join</strong> when you do not activate this option settings panel by displaying it. These options perform the join of two or more flows based on common field values. When more than one lookup tables need joining, the main input flow starts the joining from the first lookup flow, then uses the result to join the second and so on in the same manner until the last lookup flow is joined.</td>
</tr>
<tr>
<td>Join Optimization</td>
<td>None;</td>
</tr>
<tr>
<td></td>
<td>Replicated;</td>
</tr>
<tr>
<td></td>
<td>Skewed;</td>
</tr>
<tr>
<td></td>
<td>Merge.</td>
</tr>
<tr>
<td></td>
<td>The default join option is <strong>None</strong> when you do not activate this option settings panel by displaying it. These options are used to perform more efficient join operations. For example, if you are using the parallelism of multiple reduce tasks, the <strong>Skewed</strong> join can be used to counteract the load imbalance problem if the data to be processed is sufficiently skewed. Each of these options is subject to the constraints explained in Apache’s documentation about Pig Latin.</td>
</tr>
<tr>
<td>Custom Partitioner</td>
<td>Enter the Hadoop partitioner you need to use to control the partitioning of the keys of the intermediate map-outputs. For example, enter, in double quotation marks, org.apache.pig.test.utils.SimpleCustomPartitioner to use the partitioner SimpleCustomPartitioner. For further information about the code of this SimpleCustomPartitioner, see Apache’s documentation about Pig Latin. The jar file of this partitioner must have been registered in the Register jar table in the Advanced settings view of the tPigLoad component linked with the tPigMap component to be used.</td>
</tr>
</tbody>
</table>
### Lookup properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase Parallelism</td>
<td>Enter the number of reduce tasks. For further information about the parallel features, see Apache’s documentation about Pig Latin.</td>
</tr>
</tbody>
</table>

On the output side:

### Output properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catch Output Reject</td>
<td>True; False. This option, once activated, allows you to catch the records rejected by a filter you can define in the appropriate area.</td>
</tr>
<tr>
<td>Catch Lookup Inner Join Reject</td>
<td>True; False. This option, once activated, allows you to catch the records rejected by the inner join operation performed on the input flows.</td>
</tr>
</tbody>
</table>

### tPigMap Standard properties

These properties are used to configure tPigMap running in the Standard Job framework.

The Standard tPigMap component belongs to the Big Data and the Processing families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

#### Basic settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
</table>
| Mapping links display as | **Auto**: the default setting is curves links  
**Curves**: the mapping display as curves  
**Lines**: the mapping displays as straight lines. This last option allows to slightly enhance performance. |
| Map editor | It allows you to define the tPigMap routing and transformation properties. |

#### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

#### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions</td>
</tr>
</tbody>
</table>
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Possible uses are from a simple reorganization of fields to the most complex Jobs of data multiplexing or demultiplexing transformation, concatenation, inversion, filtering/splitting and more, in a Pig process.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>The use of <strong>tPigMap</strong> supposes minimum Java and Pig Latin knowledge in order to fully exploit its functionalities. This component is a junction step, and for this reason cannot be a start nor end component in the Job.</td>
</tr>
</tbody>
</table>

### Scenario: Joining data about road conditions in a Pig process

This scenario applies only to a Talend solution with Big Data.

The Job in this scenario uses two **tPigLoad** components to read data about the traffic conditions and the related events on given roads from a given Hadoop distribution, joins and filters the data using **tPigMap**, and writes the results into that Hadoop distribution using two **tPigStoreResult**.
The Hadoop distribution to be used is keeping the data about traffic situation such as normal or jam and the data about the traffic-related events such as road work, rain and even no event. In this example, the data to be used reads as follows:

1. The traffic situation data stored in the directory `/user/ychen/tpigmap/date&traffic`:

   2013-01-11 00:27:53;Bayshore Freeway;jam
   2013-02-28 07:01:18;Carpinteria Avenue;jam
   2013-01-26 11:27:59;Bayshore Freeway:normal
   2013-03-07 20:48:51;South Highway;jam
   2013-02-07 07:40:10;Lindbergh Blvd;normal
   2013-01-22 17:13:55;Pacific Hwy S;normal
   2013-03-17 23:12:26;Carpinteria Avenue;normal
   2013-01-15 08:06:53;San Diego Freeway;jam
   2013-03-19 15:18:28;Monroe Street;jam
   2013-01-20 05:53:12;Newbury Road;normal

2. The event data stored in the directory `/user/ychen/tpigmap/date&event`:

   2013-01-11 00:27:53;Bayshore Freeway;road work
   2013-02-28 07:01:18;Carpinteria Avenue;rain
   2013-01-26 11:27:59;Bayshore Freeway;road work
   2013-03-07 20:48:51;South Highway;no event
   2013-02-07 07:40:10;Lindbergh Blvd;second-hand market
   2013-01-22 17:13:55;Pacific Hwy S;no event
   2013-03-17 23:12:26;Carpinteria Avenue;no event
   2013-01-15 08:06:53;San Diego Freeway;second-hand market
   2013-03-19 15:18:28;Monroe Street;road work
   2013-01-20 05:53:12;Newbury Road;no event

For any given moment shown in the timestamps in the data, one row is logged to reflect the traffic situation and another row to reflect the traffic-related event. You need to join the data into one table in order to easily detect how the events on a given road are impacting the traffic.

**Note:**

The data used in this example is a sample with limited size.
To replicate this scenario, ensure that the Studio to be used has the appropriate right to read and write data in that Hadoop distribution and then proceed as follows:

**Linking the components**

**Procedure**

1. In the **Integration** perspective of **Talend Studio**, create an empty Job, named **pigweather** for example, from the **Job Designs** node in the **Repository** tree view.
   For further information about how to create a Job, see the **Talend Studio User Guide**.

2. Drop two **tPigLoad** components, **tPigMap** and two **tPigStoreResult** onto the workspace.
   The components can be labelled if needed be. In this scenario, we label the two **tPigLoad** components as **traffic** and **event**, respectively, which load accordingly the traffic data and the related event data. Then we label the two **tPigStoreResult** components as **normal** and **jam**, respectively, which write accordingly the results to the Hadoop distribution to be used. For further information about how to label a component, see the **Talend Studio User Guide**.

3. Right-click the **tPigLoad** component labeled **traffic** to connect it to **tPigMap** using the **Row > Pig combine** link from the contextual menu.

4. Repeat this operation to link the **tPigLoad** component labeled **event** to **tPigMap**, too. As this is the second link created, it becomes automatically the lookup link.

5. Use the **Row > Pig combine** link again to connect **tPigMap** to each of the two **tPigStoreResult** components.
   You need to name these links in the dialog box popped up once you select the link type from the contextual menu. In this scenario, we name the link to **tPigStoreResult** labeled **normal** as **out** and the link to **tPigStoreResult** labeled **jam** as **reject**.

**Configuring tPigLoad**

*Loading the traffic data*

**Procedure**

1. Double-click the **tPigLoad** labeled **traffic** to open its **Component** view.
2. Click the button next to **Edit schema** to open the schema editor.

3. Click the button three times to add three rows and in the **Column** column, rename them as **date**, **street** and **traffic**, respectively.

4. Click **OK** to validate these changes.

5. In the **Mode** area, select the **Map/Reduce** option, as we need the Studio to connect to a remote Hadoop distribution.

6. In the **Distribution** list and the **Version** field, select the Hadoop distribution to be used. In this example, it is **Hortonworks Data Platform V1.0.0**.
7. In the **Load function** list, select the **PigStorage** function to read the source data, as the data is a structured file in human-readable UTF-8 format.

8. In the **NameNode URI** and the **Resource Manager** fields, enter the locations of the master node and the Resource Manager of the Hadoop distribution to be used, respectively. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS.

9. In the **Input file URI** field, enter the directory where the data about the traffic situation is stored. As explained earlier, the directory in this example is `/user/ychen/tpigmap/date&traffic`.

10. In the **Field separator** field, enter `;` depending on the separator used by the source data.

**Loading the event data**

**Procedure**

1. Double-click the **tPigLoad** labeled **event** to open its **Component** view.

   ![Component view](image)

2. Click the ![button](image) button next to **Edit schema** to open the schema editor.

3. Click the ![button](image) button three times to add three rows and in the **Column** column, rename them as **date**, **street** and **event**, respectively.
4. Click OK to validate these changes.
5. In the Mode area, select Map/Reduce.
   As you have configured the connection to the given Hadoop distribution in that first tPigLoad
   component, traffic, this event component reuses that connection and therefore, the corresponding
   options in the Distribution and the Version lists have been automatically selected.
6. In the Load function field, select the PigStorage function to read the source data.
7. In the Input file URI field, enter the directory where the event data is stored. As explained earlier,
   the directory in this example is "/user/ychen/tpigmap/date&event".

Configuring tPigMap

1. Double-click tPigMap to open its Map Editor view.

Creating the output schema

Procedure

1. On the input side (left side) of the Map Editor, each of the two tables represents one of the input fl
ow, the upper one for the main flow and the lower one for the lookup flow.
   On the output side (right side), the two tables represent the output flows that you named as out1
   and reject earlier.
   From the main flow table, drop its three columns onto each of the output flow table.
2. From the lookup flow, drop the event column onto each of the output flow table.
   Then from the Schema editor view, you can see the schemas of the both sides have been
   completed and as well, click each table to display its schema in this view.

Setting the mapping conditions

Procedure

1. On the lookup flow table, click the button to open the setting panel in this table.
2. In the **Join Model** row, select **Left Outer Join** to ensure that every record of the main flow is included in this join.

3. On the **out1** output flow table, click the **button** to display the editing field for the filter expression.

4. Enter `'normal'== row1.traffic`
   This allows **tPigMap** to output only the traffic records reading *normal* in the **out1** flow.

5. On the **reject** output flow table, click the **button** to open the setting panel.

6. In the **Catch Output Reject** row, select **true** to output the traffic records reading *jam* in the **reject** flow.

7. Click **Apply**, then click **OK** to validate these changes and accept the propagation prompted by the pop-up dialog box.

### Configuring tPigStoreResult

**Procedure**

1. Double-click the **tPigStoreResult** labeled **normal** to open its **Component** view.

   ![Component view](image)

   2. In the **Result file** field, enter the directory you need to write the result in. In this scenario, it is `/user/ychen/tpigmap/traffic_normal`, which receives the records reading *normal*.

   3. Select **Remove result directory if exists** check box.

   4. In the **Store function** list, select **PigStorage** to write the records in human-readable UTF-8 format.

   5. In the **Field separator** field, enter `;`

   6. Repeat the same operations to configure the **tPigStoreResult** labeled **jam**, but set the directory, in the **Result file** field, as `/user/ychen/tpigmap/traffic_jam`.

**Results**

**Note:**

If either of the components does not retrieve its schema from **tPigMap**, a warning icon appears. In this case, click the **Sync columns** button to retrieve the schema from the preceding one and once done, the warning icon disappears.

### Executing the Job

Then you can press **F6** to run this Job.

Once done, verify the results in the Hadoop distribution used.
From the traffic_jam records, you can analyze what event is often going on in the meantime of a traffic jam and from the traffic_normal records, how the smooth traffic situation is maintained.

If you need to obtain more details about the Job, it is recommended to use the web console of the Jobtracker provided by the Hadoop distribution you are using.

In JobHistory, you can easily find the execution status of your Pig Job because the name of the Job is automatically created by concatenating the name of the project that contains the Job, the name and version of the Job itself and the label of the first tPigLoad component used in it. The naming convention of a Pig Job in JobHistory is ProjectName_JobNameVersion_FirstComponentName.
**tPigReplicate**

Performs different operations on the same schema.

The tPigReplicate, used after an input Pig component, duplicates the incoming schema into as many identical output flows as needed.

**tPigReplicate Standard properties**

These properties are used to configure tPigReplicate running in the Standard Job framework.

The Standard tPigReplicate component belongs to the Big Data and the Processing families.

The component in this framework is available when you are using one of the **Talend** solutions with Big Data.

**Basic settings**

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word *line* when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  • **View schema**: choose this option to view the schema only.  
  • **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.  
  Click **Sync columns** to retrieve the schema from the previous component connected in the Job. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
<td></td>
</tr>
</tbody>
</table>
### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</th>
</tr>
</thead>
</table>

### Usage

**Usage rule**

This component is not startable (green background); it requires tPigLoad as the input component and expects other Pig components to handle its output flow(s).

**Prerequisites**

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR's documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to
make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

| Connections                        | Outgoing links (from this component to another):
|                                  | **Row**: Pig combine. This link joins all data processes designed in the Job and executes them simultaneously.
|                                  | Incoming links (from one component to this one):
|                                  | **Row**: Pig combine.
|                                  | For further information regarding connections, see Talend Studio User Guide.

**Scenario: Replicating a flow and sorting two identical flows respectively**

This scenario applies only to a Talend solution with Big Data. The Job in this scenario uses Pig components to handle names and states loaded from a given HDFS system. It reads and replicates the input flow, then sorts the two identical flows based on name and state respectively, and writes the results back into that HDFS.

Before starting to replicate this Job, ensure that you have the appropriate right to read and write data in the Hadoop distribution to be used and that Pig is properly installed in that distribution.

**Linking the components**

**Procedure**

1. In the Integration perspective of Talend Studio, create an empty Job, named Replicate for example, from the Job Designs node in the Repository tree view.

   For further information about how to create a Job, see the Talend Studio User Guide.

2. Drop tPigLoad, tPigReplicate, two tPigSort and two tPigStoreResult onto the workspace.
The **tPigLoad** component reads data from the given HDFS system. The sample data used in this scenario reads as follows:

Andrew Kennedy;Mississippi  
Benjamin Carter;Louisiana  
Benjamin Monroe;West Virginia  
Bill Harrison;Tennessee  
Calvin Grant;Virginia  
Chester Harrison;Rhode Island  
Chester Hoover;Kansas  
Chester Kennedy;Maryland  
Chester Polk;Indiana  
Dwight Nixon;Nevada  
Dwight Roosevelt;Mississippi  
Franklin Grant;Nebraska

The location of the data in this scenario is `/user/ychen/raw/Name&State.csv`.

3. Connect them using the **Row > Pig combine** links.

### Configuring tPigLoad

**Procedure**

1. Double-click **tPigLoad** to open its **Component** view.

![Component view](image)

2. Click the button next to **Edit schema** to open the schema editor.
3. Click the button twice to add two rows and name them Name and State, respectively.

4. Click OK to validate these changes and accept the propagation prompted by the pop-up dialog box.

5. In the Mode area, select Map/Reduce because the Hadoop to be used in this scenario is installed in a remote machine. Once selecting it, the parameters to be set appear.

6. In the Distribution and the Version lists, select the Hadoop distribution to be used.

7. In the Load function list, select PigStorage.

8. In the NameNode URI field and the Resource Manager field, enter the locations of the NameNode and the Resource Manager to be used for Map/Reduce, respectively. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS.

9. In the Input file URI field, enter the location of the data to be read from HDFS. In this example, the location is /user/ychen/raw/NameState.csv.

10. In the Field separator field, enter the semicolon ;.

**Configuring tPigReplicate**

**Procedure**

1. Double-click tPigReplicate to open its Component view.

2. Click the button next to Edit schema to open the schema editor to verify whether its schema is identical with that of its preceding component.
Note:
If this component does not have the same schema of the preceding component, a warning icon appears. In this case, click the **Sync columns** button to retrieve the schema from the preceding one and once done, the warning icon disappears.

**Configuring tPigSort**

**About this task**

Two **tPigSort** components are used to sort the two identical output flows: one based on the *Name* column and the other on the *State* column.

**Procedure**

1. Double-click the first **tPigSort** component to open its **Component** view to define the sorting by name.

2. In the **Sort key** table, add one row by clicking the **+** button under this table.

3. In the **Column** column, select *Name* from the drop-down list and select **ASC** in the **Order** column.

4. Double-click the other **tPigSort** to open its **Component** view to define the sorting by state.
5. In the Sort key table, add one row, then select Name from the drop-down list in the Column column and select ASC in the Order column.

Configuring tPigStoreResult

About this task

Two tPigStoreResult components are used to write each of the sorted data into HDFS.

Procedure

1. Double-click either the first tPigStoreResult component to open its Component view to write the data sorted by name.

2. In the Result file field, enter the directory where the data will be written. This directory will be created if it does not exist. In this scenario, we put /user/ychen/sort/tPigreplicate/byName.csv.

3. Select Remove result directory if exists.

4. In the Store function list, select PigStorage.

5. In the Field separator field, enter the semicolon ;

6. Do the same for the other tPigStoreResult component but set another directory for the data sorted by state. In this scenario, it is /user/ychen/sort/tPigreplicate/byState.csv.
Executing the Job

About this task

Then you can run this Job.

Procedure

Press F6 to run this Job.

Results

Once done, browse to the locations where the results were written in HDFS.

The following image presents the results sorted by name:

File: /user/ychen/sort/tPigreplicate/byName.csv/part-r-00000

The following image presents the results sorted by state:
If you need to obtain more details about the Job, it is recommended to use the web console of the Jobtracker provided by the Hadoop distribution you are using.

In JobHistory, you can easily find the execution status of your Pig Job because the name of the Job is automatically created by concatenating the name of the project that contains the Job, the name and version of the Job itself and the label of the first PigLoad component used in it. The naming convention of a Pig Job in JobHistory is $\text{ProjectName\_JobNameVersion\_FirstComponentName}$. 
tPigSort

Sorts relation based on one or more defined sort keys.

**tPigSort Standard properties**

These properties are used to configure tPigSort running in the Standard Job framework.

The Standard tPigSort component belongs to the Big Data and the Processing families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

| **Built-in**: The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide.

| **Repository**: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job designs. Related topic: see Talend Studio User Guide.

| **Sort key** | Click the Add button beneath the Sort key table to add one or more lines to specify column and sorting order for each sort key.

**Advanced settings**

| **Increase parallelism** | Select this check box to set the number of reduce tasks for the MapReduce Jobs |
### tStatCatcher Statistics

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

### Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  

For further information about variables, see *Talend Studio User Guide*. |

### Usage

| Usage rule | This component is commonly used as intermediate step together with input component and output component. |

### Prerequisites

| Prerequisites | The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example.  

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).  

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.  

- Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the **[Preferences]** dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the **Data viewer** to view locally in the Studio the data stored in MapR. |
For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

**Limitation**

Knowledge of Pig scripts is required.

### Scenario: Sorting data in ascending order

This scenario applies only to a Talend solution with Big Data.

This scenario describes a three-component Job that sorts rows of data based on one or more sorting conditions and stores the result into a local file.

### Setting up the Job

**Procedure**

1. Drop the following components from the **Palette** to the design workspace: tPigSort, tPigLoad, tPigStoreResult.
2. Connect tPigLoad to tPigFilterRow using a **Row > Pig Combine** connection.
3. Connect tPigFilterRow to tPigStoreResult using a **Row > Pig Combine** connection.

### Loading the data

**Procedure**

1. Double-click **tPigLoad** to open its **Basic settings** view.
2. Click the [...] button next to Edit schema to add columns for tPigLoad.

<table>
<thead>
<tr>
<th>Column</th>
<th>Key</th>
<th>Type</th>
<th>N..</th>
<th>Date P..</th>
<th>Le..</th>
<th>Or..</th>
<th>Pr..</th>
<th>D..</th>
<th>C..</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>✓</td>
<td>String</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Age</td>
<td></td>
<td>Character</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Click the [+] button to add Name, Country and Age and click OK to save the setting.
4. Select Local from the Mode area.
5. Fill in the Input filename field with the full path to the input file.
   In this scenario, the input file is CustomerList that contains rows of names, country names and age.
6. Select PigStorage from the Load function list.
7. Leave rest of the settings as they are.

**Setting the sorting condition**

**Procedure**

1. Double-click tPigSort to open its Basic settings view.
2. Click **Sync columns** to retrieve the schema structure from the preceding component.

3. Click the [+ ] button beneath the **Sort key** table to add a new sort key. Select **Age** from the Column list and select **ASC** from the Order list.

   This sort key will sort the data in **CustomerList** in ascending order based on **Age**.

### Saving the data to a local file

**Procedure**

1. Double-click **tPigStoreResult** to open its **Basic settings** view.

2. Click **Sync columns** to retrieve the schema structure from the preceding component.

3. Select **Remove result directory if exists**.

4. Fill in the **Result file** field with the full path to the result file.

   In this scenario, the result of filter is saved in **Lucky_Customer** file.

5. Select **PigStorage** from the **Store function** list.

6. Leave rest of the settings as they are.

### Executing the Job

Save your Job and press **F6** to run it.
The *Lucky_Customer* file is generated containing the data in ascending order based on *Age*.
tPigStoreResult

Stores the result of your Pig Job into a defined data storage space.

tPigStoreResult Standard properties

These properties are used to configure tPigStoreResult running in the Standard Job framework.

The Standard tPigStoreResult component belongs to the Big Data and the Processing families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Repository or Built-in.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The Repository option allows you to reuse the connection properties centrally stored under the Hadoop cluster node of the Repository tree. Once selecting it, the button appears, then you can click it to display the list of the stored properties and from that list, select the properties you need to use. Once done, the appropriate parameters are automatically set. Otherwise, if you select Built-in, you need to manually set each of the parameters.</td>
</tr>
</tbody>
</table>

Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various
**Use S3 endpoint**

Select this check box to write data into a given Amazon S3 bucket folder.

Once this **Use S3 endpoint** check box is selected, you need to enter the following parameters in the fields that appear:

- **S3 bucket name and folder**: enter the bucket name and its folder in which you need to write data. You need to separate the bucket name and the folder name using a slash (/).
- **Access key** and **Secret key**: enter the authentication information required to connect to the Amazon S3 bucket to be used.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

Note that the format of the S3 file is S3N (S3 Native Filesystem).

**Result folder URI**

Select the path to the result file in which data is stored.

**Remove result directory if exists**

Select this check box to remove an existing result directory.

*Note:*

This check box is disabled when you select **HCatStorer** from the **Store function** list.

**Store function**

Select a store function for data to be stored:

- **PigStorage**: Stores data in UTF-8 format.
- **BinStorage**: Stores data in machine-readable format.
- **PigDump**: Stores data as tuples in human-readable UTF-8 format.
- **HCatStorer**: Stores data in HCatalog managed tables using Pig scripts.
- **HBaseStorage**: Stores data in HBase. Then you need to complete the HBase configuration in the **HBase configuration** area displayed.
- **SequenceFileStorage**: Stores data of the SequenceFile formats. Then you need to complete the configuration of the file to be stored in the **Sequence Storage Configuration** area that appears.
- **RCFilePigStorage**: Stores data of the RCFile format.
- **AvroStorage**: Stores Avro files. For further information about AvroStorage, see Apache’s documentation on [https://cwiki.apache.org/confluence/display/PIG/AvroStorage](https://cwiki.apache.org/confluence/display/PIG/AvroStorage).
- **ParquetStorer**: Stores Parquet files. Then from the **Associate tPigLoad component** list, you need...
to select the **tPigLoad** component in which the connection to the MapReduce cluster to be used is defined.

Then from the **Compression** list that appears, select the compression mode you need to use to handle the Parquet file. The default mode is **Uncompressed**.

- **Custom**: Stores data using any user-defined store function. To do this, you need to register, in the **Advanced settings** tab view, the jar file containing the function to be used, and then, in the field displayed next to this **Store function** field, specify that function.

Note that when the file format to be used is **PARQUET**, you might be prompted to find the specific Parquet jar file and install it into the Studio.

- When the connection mode to Hive is **Embedded**, the Job is run in your local machine and calls this jar installed in the Studio.

- When the connection mode to Hive is **Standalone**, the Job is run in the server hosting Hive and this jar file is sent to the HDFS system of the cluster you are connecting to. Therefore, ensure that you have properly defined the NameNode URI in the corresponding field of the **Basic settings** view.

This jar file can be downloaded from Apache's site. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### HCatalog Configuration

Fill the following fields to configure HCatalog managed tables on HDFS (Hadoop distributed file system):

**Distribution and Version**:
Select the Hadoop distribution to which you have defined the connection in the **tPigLoad** component, used in the same Pig process of the active **tPigStoreResult**.

If that **tPigLoad** component connects to a custom Hadoop distribution, you must select **Custom** for this **tPigStoreResult** component, too. Then the **Custom jar** table appears, in which, you need to add only the jar files required by the selected **Store function**.

**HCat metastore**: Enter the location of the HCatalog’s metastore, which is actually Hive’s metastore.

**Database**: The database in which tables are placed.

**Table**: The table in which data is stored.

**Partition filter**: Fill this field with the partition keys to list partitions by filter.

**Note**: **HCataLog Configuration area** is enabled only when you select **HCatStorer** from the **Store function** list. For further information about the usage of HCatalog, see [https://cwiki.apache.org/confluence/display/Hive/HCatalog](https://cwiki.apache.org/confluence/display/Hive/HCatalog). For further information about the
**HBase configuration**

This area is available to the HBaseStorage function. The parameters to be set are:

**Distribution and Version:**
Select the Hadoop distribution to which you have defined the connection in the **tPigLoad** component, used in the same Pig process of the active **tPigStoreResult**.

If that **tPigLoad** component connects to a custom Hadoop distribution, you must select **Custom** for this **tPigStoreResult** component, too. Then the **Custom jar table** appears, in which you need to add only the jar files required by the selected **Store function**.

**Zookeeper quorum:**
Type in the name or the URL of the Zookeeper service you use to coordinate the transaction between your Studio and your database. Note that when you configure the Zookeeper, you might need to explicitly set the `zookeeper.znode.parent` property to define the path to the root znode that contains all the znodes created and used by your database; then select the **Set Zookeeper znode parent** check box to define this property.

**Zookeeper client port:**
Type in the number of the client listening port of the Zookeeper service you are using.

**Table name:**
Enter the name of the HBase table you need to store data in. The table must exist in the target HBase.

**Row key column:**
Select the column used as the row key column of the HBase table.

**Store row key column to Hbase column:**
Select this check box to make the row key column an HBase column belonging to a specific column family.

**Mapping:**
Complete this table to map the columns of the table to be used with the schema columns you have defined for the data flow to be processed.

The **Column** column of this table is automatically filled once you have defined the schema; in the **Family name** column, enter the column families you want to create or use to group the columns in the **Column** column. For further information about a column family, see Apache documentation at Column families.

**Field separator:**
Enter character, string or regular expression to separate fields for the transferred data.

**Note:**

This field is enabled only when you select PigStorage from the Store function list.

### Sequence Storage configuration
This area is available only to the SequenceFileStorage function. Since a SequenceFile record consists of binary key/value pairs, the parameters to be set are:

- **Key column**: Select the Key column of a key/value record.
- **Value column**: Select the Value column of a key/value record.

### Advanced settings

#### Register jar
Click the [+] button to add rows to the table and from these rows, browse to the jar files to be added. For example, in order to register a jar file called piggybank.jar, click the [+] button once to add one row, then click this row to display the [...] browse button, and click this button to browse to the piggybank.jar file following the [Select Module] wizard.

#### HBaseStorage configuration
Add and set more HBaseStorage storer options in this table. The options are:

- **loadKey**: enter true to store the row key as the first column of the result schema, otherwise, enter false;
- **gt**: the minimum key value;
- **lt**: the maximum key value;
- **gte**: the minimum key value (included);
- **lte**: the maximum key value (included);
- **limit**: maxum number of rows to retrieve per region;
- **caching**: number of rows to cache;
- **caster**: the converter to use for writing values to HBase. For example, Utf8StorageConverter.

#### Define the jars to register
This check box appears when you are using tHCatStorer, while by default, you can leave it clear as the Studio registers the required jar files automatically. In case any jar file is missing, you can select this check box to display the Register jar for HCatalog table and set the correct path to that missing jar.

#### tStatCatcher Statistics
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component is always used to end a Pig process and needs <strong>tPigLoad</strong> at the beginning of that chain to provide data. This component reuses automatically the connection created by the <strong>tPigLoad</strong> component in that Pig process. Note that if you use <em>Hortonworks Data Platform V2.0.0</em>, the type of the operating system for running the distribution and a <em>Talend</em> Job must be the same, such as Windows or Linux. Otherwise, you have to use <em>Talend</em> Jobserver to execute the Job in the same type of operating system in which the <em>Hortonworks Data Platform V2.0.0</em> distribution you are using is run.</td>
</tr>
</tbody>
</table>

### Prerequisites

<table>
<thead>
<tr>
<th>Prerequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Hadoop distribution must be properly installed, so as to guarantee the interaction with <em>Talend Studio</em>. The following list presents MapR related information for example.</td>
</tr>
<tr>
<td>- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under <code>MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native</code>. For example, the library for Windows is <code>\lib\native\MapRClient.dll</code> in the MapR client jar file. For further information, see the following link from MapR: <a href="http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr">http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr</a>. Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.</td>
</tr>
<tr>
<td>- Set the <code>-Djava.library.path</code> argument, for example, in the <strong>Job Run VM arguments</strong> area of the <strong>Run/Debug</strong> view in the <strong>[Preferences]</strong> dialog box in the <strong>Window</strong> menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the <strong>Data viewer</strong> to view locally in the Studio the data stored in MapR.</td>
</tr>
</tbody>
</table>
For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

| Limitation | Knowledge of Pig scripts is required. If you select **HCatStorer** as the store function, knowledge of HCatLog DDL (HCatLog Data Definition Language, a subset of Hive Data Definition Language) is required. For further information about HCatLog DDL, see [https://cwiki.apache.org/confluence/display/Hive/HCatalog](https://cwiki.apache.org/confluence/display/Hive/HCatalog). |

### Related Scenario

- Related scenario in which **tPigStoreResult** uses the **Local** mode, see *Scenario: Sorting data in ascending order* on page 3003 of **tPigSort**.
- Related scenario in which **tPigStoreResult** uses the **Map/Reduce** mode, see *Scenario: Loading an HBase table* on page 2978.
tPivotToColumnsDelimited

Fine-tunes the selection of data to output.

The tPivotToColumnsDelimited outputs data based on an aggregation operation carried out on a pivot column.

**tPivotToColumnsDelimited Standard properties**

These properties are used to configure tPivotToColumnsDelimited running in the Standard Job framework.

The Standard tPivotToColumnsDelimited component belongs to the File family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pivot column</strong></td>
<td>Select the column from the incoming flow that will be used as pivot for the aggregation operation.</td>
</tr>
<tr>
<td><strong>Aggregation column</strong></td>
<td>Select the column from the incoming flow that contains the data to be aggregated.</td>
</tr>
<tr>
<td><strong>Aggregation function</strong></td>
<td>Select the function to be used in case several values are available for the pivot column.</td>
</tr>
<tr>
<td><strong>Group by</strong></td>
<td>Define the aggregation sets, the values of which will be used for calculations.</td>
</tr>
<tr>
<td><strong>Input Column</strong></td>
<td>Match the input column label with your output columns, in case the output label of the aggregation set needs to be different.</td>
</tr>
<tr>
<td><strong>File Name</strong></td>
<td>Name or path to the file to be processed and/or the variable to be used. For further information about how to define and use a variable in a Job, see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Field separator</strong></td>
<td>Character, string or regular expression to separate fields of the output file.</td>
</tr>
<tr>
<td><strong>Row separator</strong></td>
<td>String (ex: <code>\n</code> on Unix) to distinguish rows in the output file.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong></td>
<td>the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_OUT</strong></td>
<td>the number of rows written to the file by the component. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component requires an input flow.</th>
</tr>
</thead>
</table>

### Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

### Scenario: Using a pivot column to aggregate data

The following scenario describes a Job that aggregates data from a delimited input file, using a defined pivot column.

**Dropping and linking components**

**Procedure**

1. Drop the following component from the **Palette** to the design workspace: **tFileInputDelimited**, **tPivotToColumnsDelimited**.
2. Link the two components using a **Row > Main** connection.
Configuring the components

Set the input component

Procedure

1. Double-click the `tFileInputDelimited` component to open its Basic settings view.

   - Property Type: Built-In
   - File name/Stream: `D:/Input/pivot/n.csv`
   - Row Separator: ` carriage return`
   - Field Separator: `;`
   - CSV options:
     - Header: 1
     - Footer: 0
     - Limit: 0
   - Schema: Built-In

2. Browse to the input file to fill out the File Name field.
   - The file to use as input file is made of 3 columns, including: ID, Question and the corresponding Answer.

<table>
<thead>
<tr>
<th>Id</th>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Name</td>
<td>Juan</td>
</tr>
<tr>
<td>2</td>
<td>Name</td>
<td>Jean</td>
</tr>
<tr>
<td>3</td>
<td>Name</td>
<td>John</td>
</tr>
<tr>
<td>1</td>
<td>Gender</td>
<td>M</td>
</tr>
<tr>
<td>2</td>
<td>Gender</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>Gender</td>
<td>M</td>
</tr>
<tr>
<td>1</td>
<td>Surgery</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Surgery</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Surgery</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>Age</td>
<td>45</td>
</tr>
<tr>
<td>2</td>
<td>Age</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>Age</td>
<td>42</td>
</tr>
<tr>
<td>1</td>
<td>Name</td>
<td>Mary</td>
</tr>
</tbody>
</table>

3. Define the Row and Field separators, in this example, respectively: carriage return and semi-colon.
4. As the file contains a header line, define it also.
5. Set the schema describing the three columns: ID, Questions, Answers.

Set the output component

Procedure

1. Double-click the `tPivotToColumnsDelimited` component to open its Basic settings view.
2. In the **Pivot column** field, select the pivot column from the input schema. This is often the column presenting most duplicates (pivot aggregation values).

3. In the **Aggregation column** field, select the column from the input schema that should get aggregated.

4. In the **Aggregation function** field, select the function to be used in case duplicates are found out.

5. In the **Group by** table, add an Input column, that will be used to group by the aggregation column.

6. In the **File Name** field, browse to the output file path. And on the **Row** and **Field separator** fields, set the separators for the aggregated output rows and data.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.

2. Press **F6** or click **Run** on the Run tab to execute the Job.

<table>
<thead>
<tr>
<th>id</th>
<th>Name</th>
<th>Gender</th>
<th>Surgery</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mary</td>
<td>M</td>
<td>Yes</td>
<td>45</td>
</tr>
<tr>
<td>2</td>
<td>Jean</td>
<td>M</td>
<td>No</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>John</td>
<td>F</td>
<td>Yes</td>
<td>42</td>
</tr>
</tbody>
</table>

The output file shows the newly aggregated data.
tPOP

Fetches one or more email messages from a server using the POP3 or IMAP protocol.
tPOP uses the POP or IMAP protocol to connect to a specific email server. Then it fetches one or more
email messages and writes the recovered information in specified files. Parameters in the Advanced
settings view allows you to use filters on your selection.

**tPOP Standard properties**

These properties are used to configure tPOP running in the Standard Job framework.
The Standard tPOP component belongs to the Internet family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Host</th>
<th>IP address of the email server you want to connect to.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Port number of the email server.</td>
</tr>
</tbody>
</table>
| Username and Password | User authentication data for the email server.  
Username: enter the username you use to access your email box.  
Password: enter the password you use to access your email box. |
| Output directory | Enter the path to the file in which you want to store the email messages you retrieve from the email server, or click the three-dot button next to the field to browse to the file. |
| Filename pattern | Define the syntax of the names of the files that will hold each of the email messages retrieved from the email server, or press Ctrl+Space to display the list of predefined patterns. |
| Retrieve all emails? | Select this check box to retrieve all email messages present on the specified server. |
| Number of emails to retrieve | Enter the number of email messages you want to retrieve.  
This field is available only when the Retrieve all emails? check box is cleared. |
| Newer email first | Select this check box to retrieve the most recent email messages according to the number specified in the Number of emails to retrieve field, and the email messages will be returned in chronological order.  
This check box is available only when the Retrieve all emails? check box, which is selected by default, is cleared. |
### Delete emails from server

Select this check box if you do not want to keep the retrieved email messages on the server.

**Note:**
For Gmail servers, this option does not work for the POP3 protocol. Select the IMAP protocol and ensure that the Gmail account is configured to use IMAP.

### Choose the protocol

From the list, select the protocol to be used to retrieve the email messages from the server. This protocol is the one used by the email server. If you choose the IMAP protocol, you will be able to select the folder from which you want to retrieve your emails.

### Use SSL

Select this check box if your email server uses this protocol for authentication and communication confidentiality.

**Note:**
This option is obligatory for users of Gmail.

### Advanced settings

#### tStatCatcher Statistics

Select this check box to gather the job processing metadata at a job level as well as at each component level.

#### Filter

Click the plus button to add as many lines as needed to filter email messages and retrieve only a specific selection:

- **Filter item**: select one of the following filter types from the list:
  - **From**: email messages are filtered according to the sender email address.
  - **To**: email messages are filtered according to the recipient email address.
  - **Subject**: email messages are filtered according to the message subject matter.
  - **Before date**: email messages are filtered by the sending or receiving date. All messages before the set date are retrieved.
  - **After date**: email messages are filtered by the sending or receiving date. All messages after the set date are retrieved.

- **Pattern**: press Ctrl+Space to display the list of available values. Select the value to use for each filter.

#### Filter condition relation

Select the type of logical relation you want to use to combine the specified filters:

- **and**: the conditions set by the filters are combined together, the research is more restrictive.
or: the conditions set by the filters are independent, the research is large.

Global Variables

Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td>NB_EMAIL</td>
<td>the number of emails received. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>CURRENT_FILE</td>
<td>the current file name. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>CURRENT_FILEPATH</td>
<td>the current file path. This is a Flow variable and it returns a string.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component does not handle data flow, it can be used alone.</td>
<td></td>
</tr>
</tbody>
</table>

Limitation

<table>
<thead>
<tr>
<th>Limitation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>When the Use SSL check box or the imap protocol is selected, tPOP cannot work with IBM Java 6.</td>
<td></td>
</tr>
</tbody>
</table>

Scenario: Retrieving a selection of email messages from an email server

This scenario is a one-component Job that retrieves a predefined number of email messages from an email server.

- Drop the tPOP component from the Palette to the design workspace.
- Double click tPOP to display the Basic settings view and define the component properties.
- Enter the email server IP address and port number in the corresponding fields.
- Enter the username and password for your email account in the corresponding fields. In this example, the email server is called Free.
In the **Output directory** field, enter the path to the output directory manually, or click the three-dot button next to the field and browse to the output directory where the email messages retrieved from the email server are to be stored.

In the **Filename pattern** field, define the syntax you want to use to name the output files that will hold the messages retrieved from the email server, or press **Ctrl+Space** to display a list of predefined patterns. The syntax used in this example is the following: `TalendDate.getDate("yyyyMMdd-hhmmss") + "." + (counter_tPOP_1 + 1) + ".txt"`.

The output files will be stored as .txt files and are defined by date, time and arrival chronological order.

Clear the **Retrieve all emails?** field and in the **Number of emails to retrieve** field, enter the number of email messages you want to retrieve, 10 in this example.

Select the **Delete emails from server** check box to delete the email messages from the email server once they are retrieved and stored locally.

In the **Choose the protocol** field, select the protocol type you want to use. This depends on the protocol used by the email server. Certain email suppliers, like Gmail, use both protocols. In this example, the protocol used is **pop3**.

Save your Job and press **F6** to execute it.
The tPOP component retrieves the 10 recent messages from the specified email server. In the tPOP directory stored locally, a .txt file is created for each retrieved message. Each file holds the metadata of the email message headings (sender’s address, recipient’s address, subject matter) in addition to the message content.
**tPostgresPlusBulkExec**

Improves performance during Insert operations to a DB2 database.

The tPostgresPlusBulkExec executes the Insert action on the data provided.

The tPostgresplusOutputBulk and tPostgresplusBulkExec components are generally used together as part of a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tPostgresPlusOutputBulkExec component, detailed in a separate section. The advantage of using two separate components is that the data can be transformed before it is loaded in the database.

**tPostgresPlusBulkExec Standard properties**

These properties are used to configure tPostgresPlusBulkExec running in the Standard Job framework. The Standard tPostgresPlusBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components on page 598](#).

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>DB Version</th>
<th>List of database versions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Database server IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Schema</td>
<td>Name of the DB schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations: None: No operation is carried out. Drop and create table: The table is removed and created again. Create table: The table does not exist and gets created. Create table if not exists: The table is created if it does not exist. Drop table if exists and create: The table is removed if it already exists and created again. Clear table: The table content is deleted. Truncate table: The table content is deleted. You do not have the possibility to rollback the operation.</td>
</tr>
<tr>
<td>File Name</td>
<td>Name of the file to be loaded.</td>
</tr>
<tr>
<td>Warning</td>
<td>This file is located on the machine specified by the URI in the Host field so it should be on the same machine as the database server.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
</tbody>
</table>
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Built-In</th>
<th>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Action</strong></th>
<th>Select the action to be carried out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulk insert</td>
<td>Bulk update Depending on the action selected, the required information varies.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Field terminated by</strong></th>
<th>Character, string or regular expression to separate fields.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>tStat Catcher Statistics</strong></th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

### Usage

<table>
<thead>
<tr>
<th><strong>Usage rule</strong></th>
<th>This dedicated component offers performance and flexibility of DB2 query handling.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Dynamic settings</strong></th>
<th>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</th>
</tr>
</thead>
</table>
### Related scenarios

For **tPostgresPlusBulkExec** related topics, see:

- [Scenario: Inserting transformed data in MySQL database](#) on page 2537.
- [Scenario: Truncating and inserting file data into an Oracle database](#) on page 2752.
**tPostgresPlusClose**

Closes the transaction committed in the connected PostgresPlus database.

**tPostgresPlusClose Standard properties**

These properties are used to configure tPostgresPlusClose running in the Standard Job framework.

The Standard tPostgresPlusClose component belongs to the Databases family.

The component in this framework is generally available.

*Note:*

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Component list</strong></td>
<td>Select the tPostgresPlusConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **tStat Catcher Statistics** | Select this check box to collect log data at the component level. |

**Usage**

<table>
<thead>
<tr>
<th><strong>Usage rule</strong></th>
<th>This component is to be used along with PostgresPlus components, especially with tPostgresPlusConnection and tPostgresPlusCommit.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamic settings</strong></td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-</td>
</tr>
</tbody>
</table>

---
| based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

### Related scenarios

No scenario is available for the Standard version of this component yet.
**tPostgresPlusCommit**

Commits in one go a global transaction, using a unique connection, instead of doing that on every row or every batch and thus improves performance.

The tPostgresPlusCommit validates the data processed through the Job into the connected DB.

**tPostgresPlusCommit Standard properties**

These properties are used to configure tPostgresPlusCommit running in the Standard Job framework.

The Standard tPostgresPlusCommit component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tPostgresPlusConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Warning:**

*If you want to use a Row > Main connection to link tPostgresPlusCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.*

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tPostgresPlus" components, especially with the |
Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenario

For tPostgresPlusCommit related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476
tPostgresPlusConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

The tPostgresPlusConnection opens a connection to the database for a current transaction.

**tPostgresPlusConnection Standard properties**

These properties are used to configure tPostgresPlusConnection running in the Standard Job framework.

The Standard tPostgresPlusConnection component belongs to the Databases and the ELT families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>DB Version</td>
<td>List of database versions.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Schema</td>
<td>Exact name of the schema</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Enter your DB authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td>Use or register a shared DB Connection</td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the <strong>Shared DB Connection Name</strong> field displayed, enter the name for the shared database</td>
</tr>
</tbody>
</table>
connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the Use dynamic job and Use an independent process to run subjob options of the tRunJob component. Using a shared connection together with a tRunJob component with either of these two options enabled will cause your Job to fail.

Advanced settings

**Auto Commit**
Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

**tStatCatcher Statistics**
Select this check box to gather the job processing metadata at a Job level as well as at each component level.

Usage

**Usage rule**
This component is more commonly used with other tPostgresPlus* components, especially with the tPostgresPlusCommit and tPostgresPlusRollback components.

Related scenario

For tPostgresPlusConnection related scenario, see tMysqlConnection on page 2475.
tPostgresPlusInput

Executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Main row link.

The tPostgresPlusInput reads a database and extracts fields based on a query.

**tPostgresPlusInput Standard properties**

These properties are used to configure tPostgresPlusInput running in the Standard Job framework.

The component tPostgresPlusInput belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td><strong>Built-in:</strong> No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository:</strong> Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>![Database icon]</td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.</td>
</tr>
<tr>
<td>![Database icon]</td>
<td>For more information about setting up and storing database connection parameters, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>![Database icon]</td>
<td>Note: When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td>![Database icon]</td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component.</td>
</tr>
</tbody>
</table>
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>DB Version</th>
<th>List of database versions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>Exact name of the schema.</td>
</tr>
</tbody>
</table>

**Username and Password**

DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.

**Table name**

Name of the table to be read.
### Query type and Query

Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use cursor</td>
<td>When selected, helps to decide the row set to work with at a time and thus optimize performance.</td>
</tr>
<tr>
<td>Trim all the String/Char columns</td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</td>
</tr>
<tr>
<td>Trim column</td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>QUERY</td>
<td>the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
</tbody>
</table>

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible SQL queries for Postgresql databases.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The <strong>Dynamic settings</strong> table is available only when the <strong>Use an existing connection</strong> check box is selected in the <strong>Basic settings</strong> view. Once a dynamic parameter is defin</td>
</tr>
</tbody>
</table>
ed, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

For related scenarios, see:
tPostgresPlusOutput

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the job.

The tPostgresPlusOutput writes, updates, makes changes or suppresses entries in a database.

tPostgresPlusOutput Standard properties

These properties are used to configure tPostgresPlusOutput running in the Standard Job framework.

The Standard tPostgresPlusOutput component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>![ ]</td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.</td>
</tr>
<tr>
<td></td>
<td>For more information about setting up and storing database connection parameters, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>![ ]</td>
<td>Note:</td>
</tr>
<tr>
<td></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component</td>
</tr>
</tbody>
</table>
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>DB Version</th>
<th>List of database versions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Database server IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Schema</td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td>None: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td>Drop and create table: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td>Create table: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td>Create table if not exists: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td>Drop table if exists and create: The table is removed if already exists and created again.</td>
</tr>
<tr>
<td></td>
<td>Clear table: The table content is deleted.</td>
</tr>
<tr>
<td></td>
<td>Truncate table: The table content is deleted. You don't have the possibility to rollback the operation.</td>
</tr>
<tr>
<td>Action on data</td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td>Insert: Add new entries to the table. If duplicates are found, Job stops.</td>
</tr>
<tr>
<td></td>
<td>Update: Make changes to existing entries</td>
</tr>
<tr>
<td></td>
<td>Insert or update: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
</tbody>
</table>
|                  | Update or insert: Update the record with the given reference. If the record does not exist, a new record would be inserted.
Delete: Remove entries corresponding to the input flow.

Warning:

It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking Edit Schema and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the Advanced settings view where you can simultaneously define primary keys for the Update and Delete operations. To do that: Select the Use field options check box and then in the Key in update column, select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the Key in delete column for the Delete operation.

Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Die on error

This check box is selected by default. Clear the check box to skip the row on error and complete the process.
for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

**Advanced settings**

| **Commit every** | Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution. |
| **Additional Columns** | This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing. |

**Name:** Type in the name of the schema column to be altered or inserted as new column

**SQL expression:** Type in the SQL statement to be executed in order to alter or insert the relevant column data.

**Position:** Select **Before**, **Replace** or **After** following the action to be performed on the reference column.

**Reference column:** Type in a column of reference that the **tDBOutput** can use to place or replace the new or altered column.

**Use field options** | Select this check box to customize a request, especially when there is double action on data. |
| **Enable debug mode** | Select this check box to display each step during processing entries in a database. |
| **Support null in “SQL WHERE” statement** | Select this check box if you want to deal with the Null values contained in a DB table. |

**Note:**

Ensure that the Nullable check box is selected for the corresponding columns in the schema.

**Use Batch** | Select this check box to activate the batch mode for data processing. |

**Note:**

This check box is available only when you have selected the **Insert**, the **Update** or the **Delete** option in the **Action on data** field.

**Batch Size** | Specify the number of records to be processed in each batch. |

This field appears only when the **Use batch mode** check box is selected.
### tStat Catcher Statistics

Select this check box to collect log data at the component level.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_LINE_UPDATED: the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_INSERTED: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_DELETED: the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_REJECTED: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>QUERY: the query statement processed. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

### Usage

#### Usage rule

This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a PostgresPlus database. It also allows you to create a reject flow using a Row > Rejects link to filter data in error. For an example of tMySqlOutput in use, see Scenario: Retrieving data in error with a Reject link on page 2529.

#### Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defin
ed, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

For tPostgresPlusOutput related topics, see:

- Scenario: Inserting a column and altering data using tMysqlOutput on page 2521.
tPostgresPlusOutputBulk

Prepares the file to be used as parameter in the INSERT query to feed the PostgresPlus database.

The tPostgresPlusOutputBulk writes a file with columns based on the defined delimiter and the PostgresPlus standards.

The tPostgresplusOutputBulk and tPostgresplusBulkExec components are generally used together as part of a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tPostgresPlusOutputBulkExec component, detailed in a separate section. The advantage of using two separate components is that the data can be transformed before it is loaded in the database.

**tPostgresPlusOutputBulk Standard properties**

These properties are used to configure tPostgresPlusOutputBulk running in the Standard Job framework.

The Standard tPostgresPlusOutputBulk component belongs to the Databases family.

The component in this framework is generally available.

ℹ️ **Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>File Name</strong></td>
<td>Name of the file to be generated.</td>
</tr>
<tr>
<td><strong>Warning:</strong></td>
<td>This file is generated on the local machine or a shared folder on the LAN.</td>
</tr>
<tr>
<td><strong>Append</strong></td>
<td>Select this check box to add the new rows at the end of the file</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
</tbody>
</table>
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Built-In: You create and store the schema locally for this component only. Related topic: see <strong>Talend Studio User Guide.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <strong>Talend Studio User Guide.</strong></td>
</tr>
</tbody>
</table>

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### Advanced settings

| **Row separator** | String (ex: `\n` on Unix) to distinguish rows. |
| **Field separator** | Character, string or regular expression to separate fields. |
| **Include header** | Select this check box to include the column header to the file. |
| **Encoding** | Select the encoding from the list or select **Custom** and define it manually. This field is compulsory for DB data handling. |
| **tStatCatcher Statistics** | Select this check box to collect log data at the component level. |

### Global Variables

| **Global Variables** | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. |
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with tPostgresPlusBulkExec component. Used together they offer gains in performance while feeding a PostgresPlus database.</th>
</tr>
</thead>
</table>

### Related scenarios

For use cases in relation with **tPostgresPlusOutputBulk**, see the following scenarios:

- **Scenario: Inserting transformed data in MySQL database** on page 2537.
- **Scenario: Inserting data in bulk in MySQL database** on page 2544.
- **Scenario: Truncating and inserting file data into an Oracle database** on page 2752.
tPostgresPlusOutputBulkExec

Improves performance during Insert operations to a PostgresPlus database.

The tPostgresPlusOutputBulkExec executes the Insert action on the data provided.

The tPostgresplusOutputBulk and tPostgresplusBulkExec components are generally used together as part of a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tPostgresPlusOutputBulkExec component.

tPostgresPlusOutputBulkExec Standard properties

These properties are used to configure tPostgresPlusOutputBulkExec running in the Standard Job framework.

The Standard tPostgresPlusOutputBulkExec component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored.</td>
</tr>
<tr>
<td></td>
<td>The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>DB Version</td>
<td>List of database versions.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td></td>
<td>Currently, only <strong>localhost, 127.0.0.1</strong> or the exact IP address of the local machine is allowed for proper functioning. In other words, the database server must be installed on the same machine where the Studio is installed or where the Job using <strong>tPostgresPlusOutputBulkExec</strong> is deployed.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Schema</td>
<td>Exact name of the schema.</td>
</tr>
</tbody>
</table>
### Username and Password

DB user authentication data.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

### Table

Name of the table to be written. Note that only one table can be written at a time and that the table must exist for the insert operation to succeed.

### Action on table

On the table defined, you can perform one of the following operations:

- **None**: No operation is carried out.
- **Drop and create a table**: The table is removed and created again.
- **Create a table**: The table does not exist and gets created.
- **Create a table if not exists**: The table is created if it does not exist.
- **Clear a table**: The table content is deleted.

### File Name

Name of the file to be generated and loaded.

**Warning:**

This file is generated on the machine specified by the URI in the **Host** field so it should be on the same machine as the database server.

### Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.
When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

### Advanced settings

<table>
<thead>
<tr>
<th>Action</th>
<th>Select the action to be carried out</th>
</tr>
</thead>
</table>
|Bulk insert|Bulk update| Depending on the action selected, the required information varies.

<table>
<thead>
<tr>
<th>File type</th>
<th>Select the type of file being handled.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Null string</th>
<th>String displayed to indicate that the value is null.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Row separator</th>
<th>String (ex: <code>\n</code> on Unix) to distinguish rows.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Field terminated by</th>
<th>Character, string or regular expression to separate fields.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Text enclosure</th>
<th>Character used to enclose text.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>tStat Catcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is mainly used when no particular transformation is required on the data to be loaded onto the database.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Limitation</th>
<th>The database server must be installed on the same machine where the Studio is installed or where the Job using tPostgresPlusOutputBulkExec is deployed, so that the component functions properly.</th>
</tr>
</thead>
</table>

### Related scenarios

For use cases in relation with tPostgresPlusOutputBulkExec, see the following scenarios:

- Scenario: Inserting transformed data in MySQL database on page 2537.
- Scenario: Inserting data in bulk in MySQL database on page 2544.
- Scenario: Truncating and inserting file data into an Oracle database on page 2752.
tPostgresPlusRollback

Avoids to commit part of a transaction involuntarily.
The tPostgresPlusRollback cancels the transaction committed in the connected DB.

tPostgresPlusRollback Standard properties

These properties are used to configure tPostgresPlusRollback running in the Standard Job framework.
The Standard tPostgresPlusRollback component belongs to the Databases family.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tPostgresPlusConnection component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

Usage

| Usage rule | This component is more commonly used with other tPostgresPlus* components, especially with the tPostgresPlusConnection and tPostgresPlusCommit components. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |
Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

For tPostgresPlusRollback related scenario, see Scenario: Rollback from inserting data in mother/daughter tables on page 2480.
tPostgresPlusRow

Acts on the actual DB structure or on the data (although without handling data), depending on the nature of the query and the database. The SQLBuilder tool helps you write easily your SQL statements.

The tPostgresPlusRow is the specific component for the database query. It executes the SQL query stated onto the specified database. The row suffix means the component implements a flow in the job design although it doesn't provide output.

tPostgresPlusRow Standard properties

These properties are used to configure tPostgresPlusRow running in the Standard Job framework.

The Standard tPostgresPlusRow component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

Note:

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

| **DB Version** | List of database versions. |
| **Host** | Database server IP address |
| **Port** | Listening port number of DB server. |
| **Database** | Name of the database |
| **Schema** | Exact name of the schema. |
| **Username and Password** | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. |
| **Table name** | Name of the table to be read. |
| **Query type** | Either **Built-in** or **Repository**. |

**Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*. 

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*. 

---

3052
### Built-in
Fill in manually the query statement or build it graphically using SQLBuilder.

### Repository
Select the relevant query stored in the Repository. The Query field gets accordingly filled in.

#### Query
Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

#### Die on error
This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row > Rejects link.

### Advanced settings

#### Propagate QUERY's recordset
Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the use column list.

**Note:**
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY's recordset should be set to the type of Object and this component is usually followed by tParseRecordSet.

#### Use PreparedStatement
Select this check box if you want to query the database using a PreparedStatement. In the Set PreparedStatement Parameter table, define the parameters represented by “?” in the SQL instruction of the Query field in the Basic Settings tab.

- **Parameter Index:** Enter the parameter position in the SQL instruction.
- **Parameter Type:** Enter the parameter type.
- **Parameter Value:** Enter the parameter value.

**Note:**
This option is very useful if you need to execute the same query several times. Performance levels are increased.

#### Commit every
Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.

#### tStat Catcher Statistics
Select this check box to collect log data at the component level.
Global Variables

| Global Variables | QUERY: the query statement processed. This is an After variable and it returns a string. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |

Usage

| Usage rule | This component offers the flexibility of the DB query and covers all possible SQL queries. |

| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

Related scenarios

For related topics, see:

- **Scenario: Combining two flows for selective output** on page 2558
- **Procedure** on page 625.
- **Scenario: Removing and regenerating a MySQL table index** on page 2552.
tPostgresPlusSCD

Addresses Slowly Changing Dimension needs, reading regularly a source of data and logging the changes into a dedicated SCD table.

The tPostgresPlusSCD reflects and tracks changes in a dedicated MSSQL SCD table.

**tPostgresPlusSCD Standard properties**

These properties are used to configure tPostgresPlusSCD running in the Standard Job framework.

The Standard tPostgresPlusSCD component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
</tbody>
</table>
Repository: Select the Repository file where Properties are stored. The following fields are pre-filled in using fetched data.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Version</td>
<td>List of database versions.</td>
</tr>
<tr>
<td>Server</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>Name of the DB schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td>Built-in</td>
<td>The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Repository</td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>SCD Editor</td>
<td>The SCD editor helps to build and configure the data flow for slowly changing dimension outputs.</td>
</tr>
<tr>
<td></td>
<td>For more information, see SCD management methodology on page 2566.</td>
</tr>
<tr>
<td><strong>Use memory saving Mode</strong></td>
<td>Select this check box to maximize system performance.</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Source keys include Null</strong></td>
<td>Select this check box to allow the source key columns to have Null values.</td>
</tr>
<tr>
<td><strong>Warning:</strong></td>
<td>Special attention should be paid to the uniqueness of the source key(s) values when this option is selected.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>End date time details</strong></th>
<th>Specify the time value of the SCD end date time setting in the format of <strong>HH:mm:ss</strong>. The default value for this field is <strong>12:00:00</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This field appears only when SCD <strong>Type 2</strong> is used and <strong>Fixed year value</strong> is selected for creating the SCD end date.</td>
</tr>
<tr>
<td><strong>Debug mode</strong></td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>NB_LINE_UPDATED</strong>: the number of rows updated. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_REJECTED</strong>: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as Output component. It requires an Input component and Row main link as input.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the \([+]\) button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.  

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.  

For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |
| Limitation | This component does not support using SCD type 0 together with other SCD types. |

## Related scenario

For related topics, see **tMysqlSCD** on page 2563.
tPostgresPlusSCDELT

Addresses Slowly Changing Dimension needs through SQL queries (server-side processing mode), and logs the changes into a dedicated PostgresPlus SCD table.

The tPostgresPlusSCDELT reflects and tracks changes in a dedicated Oracle SCD table.

tPostgresPlusSCDELT Standard properties

These properties are used to configure tPostgresPlusSCDELT running in the Standard Job framework.

The Standard tPostgresPlusSCDELT component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally. Enter properties manually.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file where Properties are stored. The fields that come after are pre-filled in using the fetched data.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>DB Version</strong></th>
<th>List of database versions.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>The IP address of the database server.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Exact name of the schema</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>User authentication data for a dedicated database. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Source table</strong></td>
<td>Name of the input DB2 SCD table.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
</tbody>
</table>
| **Action on table** | Select to perform one of the following operations on the table defined:  
  **None**: No action carried out on the table.  
  **Drop and create table**: The table is removed and created again  
  **Create table**: A new table gets created.  
  **Create table if not exists**: A table gets created if it does not exist.  
  **Clear table**: The table content is deleted. You have the possibility to rollback the operation.  
  **Truncate table**: The table content is deleted. You don't have the possibility to rollback the operation. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  • **View schema**: choose this option to view the schema only.  
  • **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
  • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate |
the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th><strong>Built-in:</strong></th>
<th>The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Repository:</strong></td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

**Surrogate Key**  
Select the surrogate key column from the list.

**Creation**  
Select the method to be used for the surrogate key generation.

For more information regarding the creation methods, see SCD management methodology on page 2566.

**Source Keys**  
Select one or more columns to be used as keys, to ensure the unicity of incoming data.

**Use SCD Type 1 fields**  
Use type 1 if tracking changes is not necessary. SCD Type 1 should be used for typos corrections for example. Select the columns of the schema that will be checked for changes.

**Use SCD Type 2 fields**  
Use type 2 if changes need to be tracked down. SCD Type 2 should be used to trace updates for example. Select the columns of the schema that will be checked for changes.

- **Start date:** Adds a column to your SCD schema to hold the start date value. You can select one of the input schema columns as Start Date in the SCD table.

- **End Date:** Adds a column to your SCD schema to hold the end date value for the record. When the record is currently active, the End Date column shows a null value, or you can select Fixed Year value and fill it in with a fictive year to avoid having a null value in the End Date field.

- **Log Active Status:** Adds a column to your SCD schema to hold the true or false status value. This column helps to easily spot the active record.

- **Log versions:** Adds a column to your SCD schema to hold the version number of the record.

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Debug mode</strong></th>
<th>Select this check box to display each step during processing entries in a database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component is used as an output component. It requires an input component and Row main link as input. |

Dynamic settings

| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

Related Scenario

For related topics, see tMysqlSCD on page 2563.
**tPostgresqlBulkExec**

Improves performance while carrying out the Insert operations to a Postgresql database.

The tPostgresqlBulkExec executes the Insert action on the data provided.

The tPostgresqlOutputBulk and tPostgresqlBulkExec components are used together to first output the file that will be then used as parameter to execute the SQL query stated. These two steps compose the tPostgresqlOutputBulkExec component, detailed in a separate section. The interest in having two separate elements lies in the fact that it allows transformations to be carried out before the data loading in the database.

**tPostgresqlBulkExec Standard properties**

These properties are used to configure tPostgresqlBulkExec running in the Standard Job framework.

The Standard tPostgresqlBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <em>Built-in</em> or <em>Repository</em></td>
</tr>
<tr>
<td><em>Built-in</em>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><em>Repository</em>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td>1.</td>
<td>In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
</tbody>
</table>
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](#).

<table>
<thead>
<tr>
<th>DB Version</th>
<th>List of database versions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Database server IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>Name of the schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time and that the table must exist for the insert operation to succeed.</td>
</tr>
</tbody>
</table>
| Action on table | On the table defined, you can perform one of the following operations:  
   - **None**: No operation is carried out.  
   - **Drop and create table**: The table is removed and created again.  
   - **Create table**: The table does not exist and gets created.  
   - **Create table if not exists**: The table is created if it does not exist.  
   - **Drop table if exists and create**: The table is removed if it already exists and created again.  
   - **Clear table**: The table content is deleted.  
   - **Truncate table**: The table content is deleted. You do not have the possibility to rollback the operation. |
| File Name  | Name of the file to be loaded.  
   - **Warning**:  
     This file is located on the machine specified by the URI in the Host field so it should be on the same machine as the database server. |
| Schema and Edit Schema | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository** |
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### Advanced settings

<table>
<thead>
<tr>
<th>Action on data</th>
<th>On the data of the table defined, you can perform:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bulk Insert</strong></td>
<td>Add multiple entries to the table. If duplicates are found, job stops.</td>
</tr>
<tr>
<td><strong>Bulk Update</strong></td>
<td>Make simultaneous changes to multiple entries.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Copy the OID for each row</th>
<th>Retrieve the ID item for each row.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains a header line with the names of each column in the file</td>
<td>Specify that the table contains header.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File type</th>
<th>Select the type of file being handled.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null string</td>
<td>String displayed to indicate that the value is null.</td>
</tr>
<tr>
<td>Fields terminated by</td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td>Escape char</td>
<td>Character of the row to be escaped.</td>
</tr>
<tr>
<td>Text enclosure</td>
<td>Character used to enclose text.</td>
</tr>
</tbody>
</table>
Activate standard_conforming_string | Activate the variable.
---|---
Force not null for columns | Define the columns nullability. 
**Force not null**: Select the check box next to the column you want to define as not null.

**tStatCatcher Statistics** | Select this check box to collect log data at the component level.

### Usage

**Usage rule** | This component is to be used along with **tPostgresqlOut putBulk** component. Used together, they can offer gains in performance while feeding a Postgresql database.

**Dynamic settings** | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Related scenarios

For use cases in relation with **tPostgresqlBulkExec**, see the following scenarios:

- **Scenario: Inserting transformed data in MySQL database** on page 2537.
- **Scenario: Inserting data in bulk in MySQL database** on page 2544.
- **Scenario: Truncating and inserting file data into an Oracle database** on page 2752.
tPostgresqlClose

Closes the transaction committed in the connected Postgresql database.

### tPostgresqlClose Standard properties

These properties are used to configure tPostgresqlClose running in the Standard Job framework.

The Standard tPostgresqlClose component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

#### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tPostgresqlConnection</strong> component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

#### Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

#### Usage

| Usage rule | This component is to be used along with Postgresql components, especially with **tPostgresqlConnection** and **tPostgresqlCommit**. |
| Dynamic settings | Click the [+•] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-**
Related scenarios

No scenario is available for the Standard version of this component yet.
**tPostgresqlCommit**

Commits in one go a global transaction, using a unique connection, instead of doing that on every row or every batch and thus improves performance.

The tPostgresqlCommit validates the data processed through the Job into the connected DB.

**tPostgresqlCommit Standard properties**

These properties are used to configure tPostgresqlCommit running in the Standard Job framework.

The Standard tPostgresqlCommit component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tPostgresqlConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Warning:**

If you want to use a Row > Main connection to link tPostgresqlCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tPostgresql* components, especially with the |
tPostgresqlCommit and tPostgresqlRollback components.

### Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Related scenario

For tPostgresqlCommit related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476.
tPostgresqlConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

The tPostgresqlConnection opens a connection to the database for a current transaction.

**tPostgresqlConnection Standard properties**

These properties are used to configure tPostgresqlConnection running in the Standard Job framework.

The Standard tPostgresqlConnection component belongs to the Databases and the ELT families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <code>Apply</code>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <em>Built-in</em> or <em>Repository</em></td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>DB Version</strong></td>
<td>List of database versions.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Database user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Use or register a shared DB Connection</strong></td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the <strong>Shared DB Connection Name</strong> field displayed, enter the name for the shared database connection. This allows you to share one single database connection.</td>
</tr>
</tbody>
</table>
connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the Use dynamic job and Use an independent process to run subjob options of the tRunJob component. Using a shared connection together with a tRunJob component with either of these two options enabled will cause your Job to fail.

This check box is not available when the Specify a data source alias check box is selected.

Specify a data source alias

Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

This check box is not visible when the Use or register a shared DB Connection check box is selected.

Data source alias

Enter the alias of the data source created on the Talend Runtime side.

This field is available only when the Specify a data source alias check box is selected.

Advanced settings

Auto Commit

Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

tStatCatcher Statistics

Select this check box to collect log data at the component level.

Usage

Usage rule

This component is more commonly used with other tPostgresql* components, especially with
Related scenario

For `tPostgresqlConnection` related scenario, see `tMysqlConnection` on page 2475
tPostgresqlInput

Executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component via a Main row link.

The tPostgresqlInput reads a database and extracts fields based on a query.

**tPostgresqlInput Standard properties**

These properties are used to configure tPostgresqlInput running in the Standard Job framework.

The Standard tPostgresqlInput component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see *Dynamic database components* on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component <strong>Basic settings</strong> view.</td>
<td></td>
</tr>
<tr>
<td>For more information about setting up and storing database connection parameters, see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td></td>
</tr>
<tr>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
<td></td>
</tr>
<tr>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component</td>
<td></td>
</tr>
</tbody>
</table>
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

| DB Version | List of database versions. |
| Host | Database server IP address |
| Port | Listening port number of DB server. |
| Database | Name of the database |
| Schema | Exact name of the schema. |

**Username and Password**

DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

**Schema and Edit Schema**

A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

**Table name**

Name of the table to be read.
**Query type** and **Query**

Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

**Specify a data source alias**

Select this check box and specify the alias of a data source created on the **Talend Runtime** side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in **Talend Runtime**. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

This check box is not available when the **Use an existing connection** check box is selected.

**Data source alias**

Enter the alias of the data source created on the **Talend Runtime** side.

This field is available only when the **Specify a data source alias** check box is selected.

---

### Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use cursor</strong></td>
<td>When selected, helps to decide the row set to work with at a time and thus optimize performance.</td>
</tr>
<tr>
<td><strong>Trim all the String/Char columns</strong></td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</td>
</tr>
<tr>
<td><strong>Trim column</strong></td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
<td></td>
</tr>
<tr>
<td><strong>QUERY</strong>: the query statement being processed. This is a Flow variable and it returns a string.</td>
<td></td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
<td></td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible SQL queries for Postgresql databases.</th>
</tr>
</thead>
</table>

**Dynamic settings**

Click the [+\(] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

**Related scenarios**

For related scenarios, see:

- Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
tPostgresqlOutput

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the job.

The tPostgresqlOutput writes, updates, makes changes or suppresses entries in a database.

tPostgresqlOutput Standard properties

These properties are used to configure tPostgresqlOutput running in the Standard Job framework.

The Standard tPostgresqlOutput component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

![Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.](image)

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

Use an existing connection

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

Note:

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component.
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>DB Version</th>
<th>List of database versions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Database server IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Schema</td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td><strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Create table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td><strong>Create table if not exists</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop table if exists and create</strong>: The table is removed if already exists and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td></td>
<td><strong>Truncate table</strong>: The table content is deleted. You don't have the possibility to rollback the operation.</td>
</tr>
<tr>
<td>Action on data</td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Make changes to existing entries</td>
</tr>
<tr>
<td></td>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td><strong>Update or insert</strong>: Update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
</tr>
</tbody>
</table>
Delete: Remove entries corresponding to the input flow.

Warning:

It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking Edit Schema and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the Advanced settings view where you can simultaneously define primary keys for the Update and Delete operations. To do that: Select the Use field options check box and then in the Key in update column, select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the Key in delete column for the Delete operation.

Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the Repository Content window.

Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Use spatial options

Note:
This check box is available when you selected the table creation related option from the Action on table list.

Select this check box to use the PostgreSQL database as a spatial database for geographic information system (GIS). When this check box is selected, two other check boxes are displayed:

- **Create Spatial index**: select this check box to create spatial index.
- **Create geometry columns reference**: select this check box to create geometry columns reference.

For more information about the PostgreSQL database, see [http://www.postgresql.org/about/](http://www.postgresql.org/about/).

### Specify a data source alias

Select this check box and specify the alias of a data source created on the **Talend Runtime** side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in **Talend Runtime**. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

This check box is not available when the **Use an existing connection** check box is selected.

### Data source alias

Enter the alias of the data source created on the **Talend Runtime** side.

This field is available only when the Specify a data source alias check box is selected.

### Die on error

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

### Advanced settings

#### Commit every

Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.

#### Additional Columns

This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.

- **Name**: Type in the name of the schema column to be altered or inserted as new column
- **SQL expression**: Type in the SQL statement to be executed in order to alter or insert the relevant column data.
<table>
<thead>
<tr>
<th><strong>Position:</strong></th>
<th>Select <em>Before</em>, <em>Replace</em> or <em>After</em> following the action to be performed on the reference column.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reference column:</strong></td>
<td>Type in a column of reference that the <em>tDBOutput</em> can use to place or replace the new or altered column.</td>
</tr>
</tbody>
</table>
| **Use save point** | Select this check box to use savepoints in the transaction. This check box will not be available if you select:  
  - the *Die on error* check box in the *Basic settings* view, or  
  - the *Use Batch Size* check box in the *Advanced settings* view.  
  This check box will not work if you:  
  - type in 0 in the *Commit every* field, or  
  - select the *Use an existing connection* check box in the *Basic settings* view while the *Auto Commit* mode is activated in the database connection component. |
| **Use field options** | Select this check box to customize a request, especially when there is double action on data. |
| **Enable debug mode** | Select this check box to display each step during processing entries in a database. |
| **Support null in “SQL WHERE” statement** | Select this check box if you want to deal with the Null values contained in a DB table.  
  **Note:** Ensure that the Nullable check box is selected for the corresponding columns in the schema. |
| **Use Batch** | Select this check box to activate the batch mode for data processing.  
  **Note:** This check box is available only when you have selected the *Insert*, the *Update* or the *Delete* option in the *Action on data* field. |
| **Batch Size** | Specify the number of records to be processed in each batch.  
  This field appears only when the *Use batch mode* check box is selected. |
| **tStat Catcher Statistics** | Select this check box to collect log data at the component level. |

**Global Variables**

| **Global Variables** | **NB_LINE:** the number of rows processed. This is an After variable and it returns an integer. |
**NB_LINE_UPDATED**: the number of rows updated. This is an After variable and it returns an integer.

**NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.

**NB_LINE_DELETED**: the number of rows deleted. This is an After variable and it returns an integer.

**NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer.

**QUERY**: the query statement processed. This is an After variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.

---

### Usage

#### Usage rule

This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a Postgresql database. It also allows you to create a reject flow using a Row > Reverts link to filter data in error. For an example of **tMySqlOutput** in use, see Scenario: Retrieving data in error with a Reject link on page 2529.

#### Dynamic settings

Click the [+ ] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-
| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |

**Related scenarios**

For **tPostgresqlOutput** related topics, see:

- Scenario: Inserting a column and altering data using **tMysqlOutput** on page 2521.
tPostgresqlOutputBulk

Prepares the file to be used as parameters in the INSERT query to feed the Postgresql database.

The tPostgresqlOutputBulk writes a file with columns based on the defined delimiter and the Postgresql standards.

The tPostgresqlOutputBulk and tPostgresqlBulkExec components are generally used together as part of a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tPostgresqlOutputBulkExec component, detailed in a separate section. The advantage of having two separate steps is that it makes it possible to transform data before it is loaded in the database.

tPostgresqlOutputBulk Standard properties

These properties are used to configure tPostgresqlOutputBulk running in the Standard Job framework.

The Standard tPostgresqlOutputBulk component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>File Name</td>
<td>Name of the file to be generated.</td>
</tr>
<tr>
<td></td>
<td>Warning: This file is generated on the local machine or a shared folder on the LAN.</td>
</tr>
<tr>
<td>Append</td>
<td>Select this check box to add the new rows at the end of the file.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository</td>
</tr>
</tbody>
</table>
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Row separator</strong></th>
<th>String (ex: <code>\n</code>on Unix) to distinguish rows.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Field separator</strong></td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td><strong>Include header</strong></td>
<td>Select this check box to include the column header to the file.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select the encoding from the list or select <strong>Custom</strong> and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

**NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string.
variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | This component is to be used along with **tPostgresqlBulkExec** component. Used together they offer gains in performance while feeding a Postgresql database. |

### Related scenarios

For use cases in relation with **tPostgresqlOutputBulk**, see the following scenarios:

- **Scenario: Inserting transformed data in MySQL database** on page 2537.
- **Scenario: Inserting data in bulk in MySQL database** on page 2544.
- **Scenario: Truncating and inserting file data into an Oracle database** on page 2752.
tPostgresqlOutputBulkExec

Improves performance during Insert operations to a Postgresql database.

The tPostgresqlOutputBulkExec executes the Insert action on the data provided.

The tPostgresqlOutputBulk and tPostgresqlBulkExec components are generally used together as part of a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tPostgresqlOutputBulkExec component.

**tPostgresqlOutputBulkExec Standard properties**

These properties are used to configure tPostgresqlOutputBulkExec running in the Standard Job framework.

The Standard tPostgresqlOutputBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td></td>
<td>Currently, only <strong>localhost</strong>, <strong>127.0.0.1</strong> or the exact IP address of the local machine is allowed for proper functioning. In other words, the database server must be installed on the same machine where the Studio is installed or where the Job using tPostgresqlOutputBulkExec is deployed.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time and that the table must exist for the insert operation to succeed.</td>
</tr>
</tbody>
</table>
| **Action on table** | On the table defined, you can perform one of the following operations:  
- **None**: No operation is carried out.  
- **Drop and create table**: The table is removed and created again.  
- **Create table**: The table does not exist and gets created.  
- **Create table if not exists**: The table is created if it does not exist.  
- **Drop table if exists and create**: The table is removed if already exists and created again.  
- **Clear a table**: The table content is deleted. |
| **File Name** | Name of the file to be generated and loaded.  
**Warning:**  
This file is generated on the machine specified by the URI in the **Host** field and it should be on the same machine as the database server. |
| **Schema and Edit Schema** | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**.  
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.  
**Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**. |
Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

## Advanced settings

<table>
<thead>
<tr>
<th>Action on data</th>
<th>On the data of the table defined, you can perform:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulk Insert</td>
<td>Add multiple entries to the table. If duplicates are found, job stops.</td>
</tr>
<tr>
<td>Bulk Update</td>
<td>Make simultaneous changes to multiple entries.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Copy the OID for each row</th>
<th>Retrieve the ID item for each row.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Contains a header line with the names of each column in the file</th>
<th>Specify that the table contains header.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Select the encoding from the list or select CUSTOM and define it manually. This field is compulsory for DB data handling.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>File type</th>
<th>Select the type of file being handled.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Null string</th>
<th>String displayed to indicate that the value is null.</th>
</tr>
</thead>
</table>

| Row separator | String (ex: \\
 on Unix) to distinguish rows. |
|---------------|------------------------------------------|

<table>
<thead>
<tr>
<th>Fields terminated by</th>
<th>Character, string or regular expression to separate fields.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Escape char</th>
<th>Character of the row to be escaped.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Text enclosure</th>
<th>Character used to enclose text.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Activate standard_conforming_string</th>
<th>Activate the variable.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Force not null for columns</th>
<th>Define the columns nullability.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Force not null</td>
<td>Select the check box next to the column you want to define as not null.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>
**Usage**

<table>
<thead>
<tr>
<th><strong>Usage rule</strong></th>
<th>This component is mainly used when no particular transformation is required on the data to be loaded onto the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Limitation</strong></td>
<td>The database server must be installed on the same machine where the Studio is installed or where the Job using \texttt{tPostgresqlOutputBulkExec} is deployed, so that the component functions properly.</td>
</tr>
</tbody>
</table>

**Related scenarios**

For use cases in relation with \texttt{tPostgresqlOutputBulkExec}, see the following scenarios:

- Scenario: Inserting transformed data in MySQL database on page 2537.
- Scenario: Inserting data in bulk in MySQL database on page 2544.
- Scenario: Truncating and inserting file data into an Oracle database on page 2752.
tPostgresqlRollback

Avoids to commit part of a transaction involuntarily.

The tPostgresqlRollback cancels the transaction commit in the connected DB.

**tPostgresqlRollback Standard properties**

These properties are used to configure tPostgresqlRollback running in the Standard Job framework. The Standard tPostgresqlRollback component belongs to the Databases family. The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tPostgresqlConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tPostgresql* components, especially with the tPostgresqlConnection and tPostgresqlCommit components. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |
Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenario

For tPostgresqlRollback related scenario, see tMysqlRollback on page 2546
tPostgresqlRow

Acts on the actual DB structure or on the data (although without handling data), depending on the nature of the query and the database. The SQLBuilder tool helps you write easily your SQL statements.

The tPostgresqlRow is the specific component for the database query. It executes the SQL query stated onto the specified database. The row suffix means the component implements a flow in the job design although it doesn't provide output.

tPostgresqlRow Standard properties

These properties are used to configure tPostgresqlRow running in the Standard Job framework.

The Standard tPostgresqlRow component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td></td>
<td>Note: When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
</tbody>
</table>
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

| **DB Version** | List of database versions. |
| **Host** | Database server IP address. |
| **Port** | Listening port number of DB server. |
| **Database** | Name of the database. |
| **Schema** | Name of the schema. |
| **Username and Password** | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Schema and Edit Schema** | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. |
| **Query type** | Either **Built-in** or **Repository**. |

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.
### Repository
Select the relevant query stored in the Repository. The Query field gets accordingly filled in.

### Query
Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

### Specify a data source alias
Select this check box and specify the alias of a data source created on the Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503.

This check box is not available when the **Use an existing connection** check box is selected.

### Data source alias
Enter the alias of the data source created on the Talend Runtime side.

This field is available only when the **Specify a data source alias** check box is selected.

### Die on error
This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row > Rejects link.

### Advanced settings

#### Propagate QUERY’s recordset
Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the *use column* list.

**Note:**
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of Object and this component is usually followed by tParseRecordSet.

#### Use PreparedStatement
Select this check box if you want to query the database using a PreparedStatement. In the *Set PreparedStatement Parameter* table, define the parameters represented by ‘?’ in the SQL instruction of the Query field in the Basic Settings tab.

- **Parameter Index**: Enter the parameter position in the SQL instruction.
- **Parameter Type**: Enter the parameter type.
- **Parameter Value**: Enter the parameter value.

**Note:**
This option is very useful if you need to execute the same query several times. Performance levels are increased.

<table>
<thead>
<tr>
<th>Commit every</th>
<th>Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

| Global Variables | QUERY: the query statement being processed. This is a Flow variable and it returns a string. ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility benefit of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using...</td>
</tr>
</tbody>
</table>
Related scenarios

For related topics, see:

- **Scenario: Combining two flows for selective output** on page 2558
- **Procedure** on page 625.
- **Scenario: Removing and regenerating a MySQL table index** on page 2552.
tPostgresqlSCD

Addresses Slowly Changing Dimension needs, reading regularly a source of data and logging the changes into a dedicated SCD table.

The tPostgresqlSCD reflects and tracks changes in a dedicated Postgresql SCD table.

**tPostgresqlSCD Standard properties**

These properties are used to configure tPostgresqlSCD running in the Standard Job framework.

The Standard tPostgresqlSCD component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the Repository file where Properties are stored. The following fields are pre-filled in using fetched data.</td>
<td></td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Version</td>
<td>List of database versions.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>Name of the DB schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field,</td>
</tr>
<tr>
<td></td>
<td>and then in the pop-up dialog box enter the password between double</td>
</tr>
<tr>
<td></td>
<td>quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at</td>
</tr>
<tr>
<td></td>
<td>a time.</td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description. It defines the number of fields (columns)</td>
</tr>
<tr>
<td></td>
<td>to be processed and passed on to the next component. When you create a</td>
</tr>
<tr>
<td></td>
<td>Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>Click Edit schema to make changes to the schema. If the current schema is</td>
</tr>
<tr>
<td></td>
<td>of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• Change to built-in property: choose this option to change the schema to</td>
</tr>
<tr>
<td></td>
<td>Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• Update repository connection: choose this option to change the schema</td>
</tr>
<tr>
<td></td>
<td>stored in the repository and decide whether to propagate the changes to</td>
</tr>
<tr>
<td></td>
<td>all the Jobs upon completion. You can select No upon completion and</td>
</tr>
<tr>
<td></td>
<td>choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td>Built-in</td>
<td>The schema is created and stored locally for this component only.</td>
</tr>
<tr>
<td>Repository</td>
<td>The schema already exists and is stored in the Repository, hence can be</td>
</tr>
<tr>
<td>SCD Editor</td>
<td>The SCD editor helps to build and configure the data flow for slowly</td>
</tr>
<tr>
<td></td>
<td>changing dimension outputs.</td>
</tr>
<tr>
<td></td>
<td>For more information, see <em>SCD management methodology</em> on page 2566.</td>
</tr>
</tbody>
</table>
### Use memory saving Mode
Select this check box to maximize system performance.

### Source keys include Null
Select this check box to allow the source key columns to have Null values.

**Warning:**
Special attention should be paid to the uniqueness of the source key(s) values when this option is selected.

### Die on error
This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>End date time details</strong></td>
<td>Specify the time value of the SCD end date time setting in the format of <strong>HH:mm:ss</strong>. The default value for this field is <strong>12:00:00</strong>.</td>
</tr>
<tr>
<td></td>
<td>This field appears only when SCD Type 2 is used and Fixed year value is selected for creating the SCD end date.</td>
</tr>
<tr>
<td><strong>Debug mode</strong></td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE_UPDATED</strong>: the number of rows updated. This is an After variable and it returns an integer.</td>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_REJECTED</strong>: the number of rows rejected. This is an After variable and it returns an integer.</td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

---

**NB_LINE_UPDATED**: the number of rows updated. This is an After variable and it returns an integer.

**NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.

**NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*. 
## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as Output component. It requires an Input component and Row main link as input.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [+]-button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.  

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |
| Limitation | This component does not support using SCD type 0 together with other SCD types. |

## Related scenario

For related topics, see tMysqlSCD on page 2563.
tPostgresqlSCDELT

Addresses Slowly Changing Dimension needs through SQL queries (server-side processing mode), and logs the changes into a dedicated DB2 SCD table.

The tPostgresqlSCDELT reflects and tracks changes in a dedicated Postgresql SCD table.

**tPostgresqlSCDELT Standard properties**

These properties are used to configure tPostgresqlSCDELT running in the Standard Job framework.

The Standard tPostgresqlSCDELT component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally. Enter properties manually.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file where Properties are stored. The fields that come after are pre-filled in using the fetched data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
</tbody>
</table>
For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](#).

<table>
<thead>
<tr>
<th><strong>DB Version</strong></th>
<th>List of database versions.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>The IP address of the database server.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>User authentication data for a dedicated database. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Source table</strong></td>
<td>Name of the input DB2 SCD table.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
</tbody>
</table>
| **Action on table** | Select to perform one of the following operations on the table defined:  
  - **None**: No action carried out on the table.  
  - **Drop and create table**: The table is removed and created again  
  - **Create table**: A new table gets created.  
  - **Create table if not exists**: A table gets created if it does not exist.  
  - **Clear table**: The table content is deleted. You have the possibility to rollback the operation.  
  - **Truncate table**: The table content is deleted. You do not have the possibility to rollback the operation. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
  Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No |
upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

### Surrogate Key
Select the surrogate key column from the list.

### Creation
Select the method to be used for the surrogate key generation.

For more information regarding the creation methods, see *SCD management methodology* on page 2566.

### Source Keys
Select one or more columns to be used as keys, to ensure the unicity of incoming data.

### Use SCD Type 1 fields
Use type 1 if tracking changes is not necessary. SCD Type 1 should be used for typos corrections for example. Select the columns of the schema that will be checked for changes.

### Use SCD Type 2 fields
Use type 2 if changes need to be tracked down. SCD Type 2 should be used to trace updates for example. Select the columns of the schema that will be checked for changes.

- **Start date**: Adds a column to your SCD schema to hold the start date value. You can select one of the input schema columns as Start Date in the SCD table.

- **End Date**: Adds a column to your SCD schema to hold the end date value for the record. When the record is currently active, the End Date column shows a null value, or you can select *Fixed Year value* and fill it in with a fictive year to avoid having a null value in the End Date field.

- **Log Active Status**: Adds a column to your SCD schema to hold the true or false status value. This column helps to easily spot the active record.

- **Log versions**: Adds a column to your SCD schema to hold the version number of the record.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Debug mode</strong></th>
<th>Select this check box to display each step during processing entries in a database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
## Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#). |

## Usage

| Usage rule | This component is used as an output component. It requires an input component and Row main link as input. |

| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |

## Related Scenario

For related topics, see tMysqlSCD on page 2563.
tPostjob

Triggers a task required after the execution of a Job

BoxPostjob starts the execution of a postjob.

tPostjob Standard properties

These properties are used to configure tPostjob running in the Standard Job framework.
The Standard tPostjob component belongs to the Orchestration family.
The component in this framework is generally available.

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by
the component when an error occurs. This is an After
variable and it returns a string. This variable functions
only if the Die on error check box is cleared, if the
component has this check box.

A Flow variable functions during the execution of
a component while an After variable functions after
the execution of the component.

To fill up a field or expression with a variable, press
Ctrl + Space to access the variable list and choose the
variable to use from it.

For further information about variables, see Talend
Studio User Guide. |

Usage

| Usage rule | tPostjob is a start component and can only be used with
an iterate connection to the next component. |

Connections

| Outgoing links (from this component to another):
| Trigger: On Component Ok. |

| Incoming links (from one component to this one):
| Trigger: Synchronize; Parallelize. |

For further information regarding connections, see
Talend Studio User Guide.

For more information about the tPostjob component, see Talend Studio User Guide.

Related scenarios

- Scenario: Handling files before and after the execution of a data Job on page 3109
- Scenario: Creating a bar chart from the input data on page 334
tPrejob

Triggers a task required for the execution of a Job

TPrejob starts the execution of a prejob.

**tPrejob Standard properties**

These properties are used to configure tPrejob running in the Standard Job framework.
The Standard tPrejob component belongs to the Orchestration family.
The component in this framework is generally available.

**Global Variables**

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide*. |

**Usage**

| Usage rule | tPrejob is a start component and can only be used with an iterate connection to the next component. |

| Connections | Outgoing links (from this component to another): **Trigger**: On Component Ok. 

**Incoming links (from one component to this one):**

**Trigger**: Synchronize; Parallelize.

For further information regarding connections, see *Talend Studio User Guide*. |

For more information about the **tPrejob** component, see *Talend Studio User Guide*. |
**Scenario: Handling files before and after the execution of a data Job**

In this scenario, a pre-job and a post-job are added to the Job describes in *Scenario 2: Finding duplicate files between two folders* on page 1101 to handle files before and after the execution of the main data Job.

As described in the above-mentioned scenario, when the main data Job is started, it creates a temporary text file to store the fetched filenames. If the specified temporary file already exists, it will be overwritten.

To prevent possible loss of data, the pre-job saves a backup copy of the specified file by renaming it, if it exists, before the main Job is executed.

After the execution of the main data Job, the post-job deletes the temporary file created by the data Job and restores the backup file, if any.

---

**Adding and linking components**

**Procedure**

1. With the main Job open on the design workspace, add a **tPrejob**, a **tPostjob**, a **tFileDelete**, and two **tFileCopy** components to the Job.
2. Link the **tPrejob** component to the first **tFileCopy** component using a **Trigger > On Component Ok** connection to build the pre-job.
3. Link the **tPostjob** component to the **tFileDelete** component using a **Trigger > On Component Ok** connection, and link the **tFileDelete** component to the other **tFileCopy** component to build the post-job.
4. Label the relevant components to better identify their roles.
Configuring the components

Configuring the pre-job

About this task
In the pre-job, the tPrejob component does not have any parameter or property to configure; it just triggers the tFileCopy component before the execution of the main data Job to rename the specified file, so all the required configurations are made in the tFileCopy component.

Procedure
1. Double-click the tFileCopy component to show its Basic settings view.

   ![Save backup (tFileCopy_1)](image)

   2. Fill the File Name field with the path and filename of the temporary text file to be renamed, D:/temp/tempdata.csv in this example.

   3. In the Destination directory field, specify or browse to destination directory. In this example, we will save the backup copy in the same directory, D:/temp/.

   4. Select the Rename check box, and specify the new filename in the Destination filename field, backup-tempdata.csv. Leave the other parameters as they are.

Configuring the post-job

About this task
In the post-job, the tPostjob component does not have any parameter or property to configure; it just triggers the tFileDelete component after the execution of the main data Job to delete the temporary file used to store the fetched filenames, which then triggers the tFileCopy component to name the backup file back to its original name.

Procedure
1. In the Basic settings view of the tFileDelete component, fill the File Name field with the path and filename of the temporary file to be deleted, D:/temp/tempdata.csv in this example, and leave the other parameters as they are.
2. Double-click the tFileCopy component to open its Basic settings view.

3. Fill the File Name field with the path and filename of the backup file, D:/temp/backup-tempdata.csv in this example.

4. In the Destination directory field, specify or browse to destination directory, D:/temp/ in this example.

5. Select the Rename check box, and specify the original name of the temporary file in the Destination filename field, tempdata.csv.

6. Select the Remove source file check box to remove the backup file after the renaming action. Leave the other parameters as they are.

**Expected execution result**

If the specified temporary file exists, its data will be saved in a backup copy before the main data Job is executed and restored thereafter.

If the temporary file does not exist, the two tFileCopy components will generate an error, but this does not prevent the main data Job from being executed.

For the execution result of the main data Job, see Scenario 2: Finding duplicate files between two folders on page 1101.

**Related scenario**

For another scenario that uses the tPrejob component, see Scenario: Creating a bar chart from the input data on page 334.
tPubSubOutput

Receives messages serialized into byte arrays by its preceding component and issues these messages into a given PubSub service.
**tRedshiftBulkExec**

Loads data into Amazon Redshift from Amazon S3, Amazon EMR cluster, Amazon DynamoDB, or remote hosts.

The `tRedshiftOutputBulk` and `tRedshiftBulkExec` components can be used together in a two step process to load data to Amazon Redshift from a delimited/CSV file on Amazon S3. In the first step, a delimited/CSV file is generated. In the second step, this file is used in the INSERT statement used to feed Amazon Redshift. These two steps are fused together in the `tRedshiftOutputBulkExec` component. The advantage of using two separate steps is that the data can be transformed before it is loaded to Amazon Redshift.

`tRedshiftBulkExec` loads data into an Amazon Redshift table from an Amazon DynamoDB table or from data files located in an Amazon S3 bucket, an Amazon EMR cluster, or a remote host that is accessed using an SSH connection.

**tRedshiftBulkExec Standard properties**

These properties are used to configure `tRedshiftBulkExec` running in the Standard Job framework.

The Standard `tRedshiftBulkExec` component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property Type</strong></td>
<td>Either <strong>Built-In</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: Select the repository file where the properties are stored.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Type in the IP address or hostname of the database server.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Type in the listening port number of the database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Type in the name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Type in the name of the schema.</td>
</tr>
</tbody>
</table>
**Username and Password**  
Type in the database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**Additional JDBC Parameters**  
Specify additional JDBC properties for the connection you are creating. The properties are separated by ampersand & and each property is a key-value pair. For example, `ssl=true & sslfactory=com.amazon.redshift.ssl.N onValidatingFactory`, which means the connection will be created using SSL.

**Table Name**  
Specify the name of the table to be written. Note that only one table can be written at a time.

**Action on table**  
On the table defined, you can perform one of the following operations:

- **None**: No operation is carried out.
- **Drop and create table**: The table is removed and created again.
- **Create table**: The table does not exist and gets created.
- **Create table if not exists**: The table is created if it does not exist.
- **Drop table if exists and create**: The table is removed if it already exists and created again.
- **Clear table**: The table content is deleted. You have the possibility to rollback the operation.

**Schema and Edit schema**  
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the
Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

| Data source type | Select the location of the source data to be loaded.  
|------------------|------------------------------------------------------|
| • **S3**: load data from a file in an Amazon S3 bucket.  
| • **EMR**: load data from an Amazon EMR cluster.  
| • **DynamoDB**: load data from an existing DynamoDB table.  
| • **Remote host**: load data from one or more remote hosts, such as Amazon Elastic Compute Cloud (Amazon EC2) instances or other computers.  
| For more information, see Data Sources. |

| Access Key | Specify the Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit Getting Your AWS Access Keys. |

| Secret Key | Specify the Secret Access Key, constituting the security credentials in combination with the access Key.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
<td></td>
</tr>
</tbody>
</table>

| Bucket | Specify the name of the Amazon S3 bucket in which the file is located.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>This field is available only when <strong>S3</strong> or <strong>Remote host</strong> is selected from the Data source type drop-down list.</td>
<td></td>
</tr>
</tbody>
</table>

| Key | Specify the object key for the file that contains the data to be loaded.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>This field is available only when <strong>S3</strong> is selected from the Data source type drop-down list.</td>
<td></td>
</tr>
</tbody>
</table>

| Cluster id | Specify the ID of the cluster that stores the data to be loaded.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>This field is available only when <strong>EMR</strong> is selected from the Data source type drop-down list.</td>
<td></td>
</tr>
</tbody>
</table>

| HDFS path | Specify the HDFS file path that references the data file.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>This field is available only when <strong>EMR</strong> is selected from the Data source type drop-down list.</td>
<td></td>
</tr>
</tbody>
</table>

| Table | Specify the name of the DynamoDB table that contains the data to be loaded.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>This field is available only when <strong>DynamoDB</strong> is selected from the Data source type drop-down list.</td>
<td></td>
</tr>
</tbody>
</table>

| Read ratio | Specify the percentage of the DynamoDB table’s provisioned throughput to use for the data load. |
### Advanced settings

<table>
<thead>
<tr>
<th><strong>File type</strong></th>
<th>Select the type of the file that contains the data to be loaded.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Delimited file or CSV</strong>: a delimited/CSV file.</td>
</tr>
<tr>
<td></td>
<td>• <strong>JSON</strong>: a JSON file.</td>
</tr>
<tr>
<td></td>
<td>• <strong>AVRO</strong>: an Avro file.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Fixed width</strong>: a fixed-width file.</td>
</tr>
<tr>
<td></td>
<td>This list is available when <strong>S3</strong>, <strong>EMR</strong>, or <strong>Remote host</strong> is selected from the <strong>Data source type</strong> drop-down list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Fields terminated by</strong></th>
<th>Enter the character used to separate fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This field is available only when <strong>Delimited file or CSV</strong> is selected from the <strong>File type</strong> list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Enclosed by</strong></th>
<th>Select the character in which the fields are enclosed.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This list is available only when <strong>Delimited file or CSV</strong> is selected from the <strong>File type</strong> list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>JSON mapping</strong></th>
<th>Specify how to map the data elements in the source file to the columns in the target table on Amazon Redshift. The valid values are:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>auto</strong>: Map the data by matching object keys or names in the source name/value pairs for a JSON file or field names in the Avro schema for an Avro file to the names of columns in the target table. The argument is case-sensitive and must be enclosed in double quotation marks.</td>
</tr>
<tr>
<td></td>
<td>• <strong>s3://jsonpaths_file</strong>: Map the data using the named JSONPaths file. The parameter must be an Amazon S3 object key that is enclosed in double quotation marks and explicitly references a single file, for example, <strong>s3://mybucket/jsonpaths.txt</strong>. For more information, see <strong>Data Format Parameters</strong>.</td>
</tr>
<tr>
<td></td>
<td>This field is available only when <strong>JSON</strong> or <strong>AVRO</strong> is selected from the <strong>File type</strong> list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Fixed width mapping</strong></th>
<th>Enter a string that specifies a user-defined column label and column width between double quotation marks. The format of the string is:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>**ColumnLabel1:**ColumnWidth1,<strong>ColumnLabel2:<strong>ColumnWidth2,</strong>...,</strong></td>
</tr>
</tbody>
</table>

This field is available only when **DynamoDB** is selected from the **Data source type** drop-down list.

Specify the object key for the SSH manifest file that provides the information used to open SSH connections and execute remote commands. This field is available only when **Remote host** is selected from the **Data source type** drop-down list.
<table>
<thead>
<tr>
<th><strong>Compressed by</strong></th>
<th>Select this check box and from the list displayed select the compression type of the source file. This check box is available when S3, EMR, or Remote host is selected from the Data source type drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Encrypt</strong></td>
<td>Select this check box if the file is encrypted using Amazon S3 client-side encryption. In the Encryption key field displayed, specify the encryption key used to encrypt the file. Note that only a base64 encoded AES 128-bit or AES 256-bit envelope key is supported. For more information, see Loading Encrypted Data Files from Amazon S3. This check box is available when S3, EMR, or Remote host is selected from the Data source type drop-down list.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select the encoding type of the data to be loaded from the list. This list is available when S3, EMR, or Remote host is selected from the Data source type drop-down list.</td>
</tr>
<tr>
<td><strong>Date format</strong></td>
<td>Select one of the following items from the list to specify the date format in the source data: • <strong>NONE</strong>: No date format is specified. • <strong>PATTERN</strong>: Select this item and specify the date format in the field displayed. The default date format is YYYY-MM-DD. • <strong>AUTO</strong>: Select this item if you want Amazon Redshift to recognize and convert automatically the date format.</td>
</tr>
<tr>
<td><strong>Time format</strong></td>
<td>Select one of the following items from the list to specify the time format in the source data: • <strong>NONE</strong>: No time format is specified. • <strong>PATTERN</strong>: Select this item and specify the time format in the field displayed. The default time format is YYYY-MM-DD HH:MI:SS. • <strong>AUTO</strong>: Select this item if you want Amazon Redshift to recognize and convert automatically the time format. • <strong>EPOCHSECS</strong>: Select this item if the source data is represented as epoch time, the number of seconds since Jan 1, 1970 00:00:00 UTC.</td>
</tr>
</tbody>
</table>

Note that the column label in the string has no relation to the table column name and it can be either a text string or an integer. The order of the label/width pairs must match the order of the table columns exactly. This field is available only when Fixed width is selected from the File type list.
- **EPOCHMILLISECS**: Select this item if the source data is represented as epoch time, the number of milliseconds since Jan 1, 1970 00:00:00 UTC.

**Settings**

Click the [*] button below the table to specify more parameters for loading the data.

- **Parameter**: Click the cell and select a parameter from the drop-down list.
- **Value**: Set the value for the corresponding parameter. Note that you cannot set the value for a parameter (such as `IGNOREBLANKLINES`) that does not need a value.

For more information about the parameters, see [http://docs.aws.amazon.com/redshift/latest/dg/r_Copy.html](http://docs.aws.amazon.com/redshift/latest/dg/r_Copy.html).

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](http://docs.aws.amazon.com/redshift/latest/dg/r_Copy.html).

**Usage**

**Usage rule**

The `tRedshiftBulkExec` component supports loading data to Amazon Redshift from a delimited/CSV, JSON, or fixed-width file on Amazon S3, but the `tRedshiftOutputBulk` component now only supports generating and uploading a delimited/CSV file to Amazon S3. When you need to load data from a JSON or fixed-width file, you can use the `tFileOutputJSON` or `tFileOutputPositional` component together with the `tS3Put` component instead of using the `tRedshiftOutputBulk` component to generate and upload the file to Amazon S3.

**Dynamic settings**

Click the [*] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases,
especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

---

**Scenario: loading/unloading data from/to Amazon S3**

This scenario describes a Job that generates a delimited file and uploads the file to S3, loads data from the file on S3 to Redshift and displays the data on the console, then unloads the data from Redshift to files on S3 per slice of the Redshift cluster, and finally lists and gets the unloaded files on S3.
Prerequisites:
The following context variables have been created and saved in the Repository tree view. For more information about context variables, see Talend Studio User Guide.

- **redshift_host**: the connection endpoint URL of the Redshift cluster.
- **redshift_port**: the listening port number of the database server.
- **redshift_database**: the name of the database.
- **redshift_username**: the username for the database authentication.
- **redshift_password**: the password for the database authentication.
- **redshift_schema**: the name of the schema.
- **s3_accesskey**: the access key for accessing Amazon S3.
- **s3_secretkey**: the secret key for accessing Amazon S3.
- **s3_bucket**: the name of the Amazon S3 bucket.
Adding and linking components

Procedure

1. Create a new Job and apply all context variables listed above to the new Job.

2. Add the following components by typing their names in the design workspace or dropping them from the Palette: a `tRowGenerator` component, a `tRedshiftOutputBulk` component, a `tRedshiftBulkExec` component, a `tRedshiftInput` component, a `tLogRow` component, a `tRedshiftUnload` component, a `tS3List` component, and a `tS3Get` component.

3. Link `tRowGenerator` to `tRedshiftOutputBulk` using a `Row > Main` connection.

4. Do the same to link `tRedshiftInput` to `tLogRow`.

5. Link `tS3List` to `tS3Get` using a `Row > Iterate` connection.

6. Link `tRowGenerator` to `tRedshiftBulkExec` using a `Trigger > On Subjob Ok` connection.

7. Do the same to link `tRedshiftBulkExec` to `tRedshiftInput`, link `tRedshiftInput` to `tRedshiftUnload`, link `tRedshiftUnload` to `tS3List`.

Configuring the components

Preparing a file and uploading the file to S3

Procedure

1. Double-click `tRowGenerator` to open its `RowGenerator Editor`.

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Type</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>redshift_host</td>
<td>String</td>
<td>redshift-cluster.abc123.redshift.amazonaws.com</td>
</tr>
<tr>
<td>2</td>
<td>redshift_port</td>
<td>String</td>
<td>5439</td>
</tr>
<tr>
<td>3</td>
<td>redshift_database</td>
<td>String</td>
<td>redshiftdb</td>
</tr>
<tr>
<td>4</td>
<td>redshift_username</td>
<td>String</td>
<td>talend</td>
</tr>
<tr>
<td>5</td>
<td>redshift_password</td>
<td>Password</td>
<td>************</td>
</tr>
<tr>
<td>6</td>
<td>redshift_schema</td>
<td>String</td>
<td>redshiftschema</td>
</tr>
<tr>
<td>7</td>
<td>s3_accesskey</td>
<td>String</td>
<td>ABCDEFGHIJKLMNOPQRSTUVWXYZ</td>
</tr>
<tr>
<td>8</td>
<td>s3_secretkey</td>
<td>String</td>
<td>a81cDeF2gHjJk=12mNoPqSt+hs+h7+6ypvDGT5V4U4</td>
</tr>
<tr>
<td>9</td>
<td>s3_bucket</td>
<td>String</td>
<td>redshift-bucket-talend</td>
</tr>
</tbody>
</table>
2. Click the [+] button to add two columns: **ID** of Integer type and **Name** of String type.

3. Click the cell in the **Functions** column and select a function from the list for each column. In this example, select **Numeric.sequence** to generate sequence numbers for the **ID** column and select **TalendDataGenerator.getFirstName** to generate random first names for the **Name** column.

4. In the **Number of Rows for RowGenerator** field, enter the number of data rows to generate. In this example, it is 20.

5. Click **OK** to close the schema editor and accept the propagation prompted by the pop-up dialog box.

6. Double-click **tRedshiftOutputBulk** to open its **Basic settings** view on the **Component** tab.

7. In the **Data file path at local** field, specify the local path for the file to be generated. In this example, it is `E:/Redshift/redshift_bulk.txt`.

8. In the **Access Key** field, press **Ctrl + Space** and from the list select `context.s3_accesskey` to fill in this field.

   Do the same to fill the **Secret Key** field with `context.s3_accesskey` and the **Bucket** field with `context.s3_bucket`.

9. In the **Key** field, enter a new name for the file to be generated after being uploaded on Amazon S3. In this example, it is **person_load**.
Loading data from the file on S3 to Redshift

Procedure

1. Double-click tRedshiftBulkExec to open its Basic settings view on the Component tab.

2. In the Host field, press Ctrl + Space and from the list select context.redshift_host to fill in this field. Do the same to fill:
   - the Port field with context.redshift_port,
   - the Database field with context.redshift_database,
   - the Schema field with context.redshift_schema,
   - the Username field with context.redshift_username,
   - the Password field with context.redshift_password,
   - the Access Key field with context.s3_accesskey,
   - the Secret Key field with context.s3_secretkey, and
   - the Bucket field with context.s3_bucket.

3. In the Table Name field, enter the name of the table to be written. In this example, it is person.

4. From the Action on table list, select Drop table if exists and create.

5. In the Key field, enter the name of the file on Amazon S3 to be loaded. In this example, it is person_load.

6. Click the [...] button next to Edit schema and in the pop-up window define the schema by adding two columns: ID of Integer type and Name of String type.
Retrieving data from the table on Redshift

Procedure

1. Double-click tRedshiftInput to open its Basic settings view on the Component tab.

2. Fill the Host, Port, Database, Schema, Username, and Password fields with their corresponding context variables.

3. In the Table Name field, enter the name of the table to be read. In this example, it is person.

4. Click the [...] button next to Edit schema and in the pop-up window define the schema by adding two columns: ID of Integer type and Name of String type.

5. In the Query field, enter the following SQL statement based on which the data are retrieved.

   "SELECT * FROM " + context.redshift_schema + "person ORDER BY "ID""

6. Double-click tLogRow to open its Basic settings view on the Component tab.
7. In the Mode area, select Table (print values in cells of a table) for a better display of the result.

**Unloading data from Redshift to file(s) on S3**

**Procedure**

1. Double-click `tRedshiftUnload` to open its Basic settings view on the Component tab.

2. Fill the Host, Port, Database, Schema, Username, and Password fields with their corresponding context variables.

3. Fill the Access Key, Secret Key, and Bucket fields also with their corresponding context variables.

4. In the Table Name field, enter the name of the table from which the data will be read. In this example, it is `person`.

5. Click the [...] button next to Edit schema and in the pop-up window define the schema by adding two columns: ID of Integer type and Name of String type.

6. In the Query field, enter the following SQL statement based on which the result will be unloaded.

   ```sql
   "SELECT * FROM person"
   ```

7. In the Key prefix field, enter the name prefix for the unload files. In this example, it is `person_unload_`. 
Retrieving files unloaded to Amazon S3

Procedure

1. Double-click tS3List to open its Basic settings view on the Component tab.

   ![tS3List_1](image)

   - Connection
     - Use an existing connection
     - Access Key: `context.s3_accesskey`
     - Secret Key: `context.s3_secretkey`
     - Inherit credentials from AWS role
     - Assume Role
   - Region: `US Standard`
   - List all buckets objects
   - Bucket:
     - Bucket name: `context.s3_bucket`
     - Key prefix: "person_unload_"

2. Fill the Access Key and Secret Key fields with their corresponding context variables.
3. From the Region list, select the AWS region where the unload files are created. In this example, it is US Standard.
4. Clear the List all buckets objects check box, and click the [+ ] button under the table displayed to add one row.
   - Fill in the Bucket name column with the name of the bucket in which the unload files are created. In this example, it is the context variable `context.s3_bucket`.
   - Fill in the Key prefix column with the name prefix for the unload files. In this example, it is `person_unload_`.

5. Double-click tS3Get to open its Basic settings view on the Component tab.

   ![tS3Get_1](image)

   - Connection
     - Use an existing connection
     - Access Key: `context.s3_accesskey`
     - Secret Key: `context.s3_secretkey`
     - Inherit credentials from AWS role
     - Assume Role
   - Region: `US Standard`
   - Encrypt
   - Bucket: `context.s3_bucket`
   - Key: `"E\//Redshift/\" + ((String)globalMap.get("S3List\_1\_CURRENT\_KEY"))`
   - File: `"E\//Redshift/\" + ((String)globalMap.get("S3List\_1\_CURRENT\_KEY"))`

6. Fill the Access Key field and Secret Key field with their corresponding context variables.
7. From the Region list, select the AWS region where the unload files are created. In this example, it is US Standard.
8. In the **Bucket** field, enter the name of the bucket in which the unload files are created. In this example, it is the context variable `context.s3_bucket`.

   In the **Key** field, enter the name of the unload files by pressing **Ctrl + Space** and from the list selecting the global variable `((String)globalMap.get("tS3List_1_CURRENT_KEY"))`.

9. In the **File** field, enter the local path where the unload files are saved. In this example, it is `E:/Redshift/ + ((String)globalMap.get("tS3List_1_CURRENT_KEY"))`.

### Saving and executing the Job

**Procedure**

1. Press **Ctrl + S** to save the Job.

2. Execute the Job by pressing **F6** or clicking **Run** on the **Run** tab.

   ![Image of save and execute process](image.png)

   ```plaintext
   [statistics] connecting to socket on port 3726
   [statistics] connected
   tLogRow_1
   ID  Name
   1  Abraham
   2  Herbert
   3  Chester
   4  John
   5  Gerald
   6  Lyndon
   7  Warren
   8  Calvin
   9  Chester
   10  Herbert
   11  Ronald
   12  Herbert
   13  Martin
   14  James
   15  Franklin
   16  Herbert
   17  Richard
   18  Warren
   19  Jimmy
   20  George
   [statistics] disconnected
   ```
As shown above, the generated data is written into the local file `redshift_bulk.txt`, the file is uploaded on S3 with the new name `person_load`, and then the data is loaded from the file on S3 to the table `person` in Redshift and displayed on the console. After that, the data is unloaded from the table `person` in Redshift to two files `personUnload_0000_part_00` and `personUnload_0001_part_00` on S3 per slice of the Redshift cluster, and finally the unloaded files on S3 are listed and retrieved in the local folder.
tRedshiftClose

Closes the transaction committed in the connected DB.

This component is used together with tRedshiftConnection and tRedshiftCommit to ensure the integrity of the transaction performed into the database.

**tRedshiftClose Standard properties**

These properties are used to configure tRedshiftClose running in the Standard Job framework.

The Standard tRedshiftClose component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tRedshiftConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with Amazon Redshift components, especially with tRedshiftConnection and tRedshiftCommit.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.</td>
</tr>
</tbody>
</table>
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Related scenario

For a related scenario, see Scenario: Handling data with Redshift on page 3139.
tRedshiftCommit

Provides gain in performance.

Using a unique connection, this component commits in one go a global transaction instead of doing that on every row or every batch.

tRedshiftCommit validates the data processed through the Job into the connected database.

**tRedshiftCommit Standard properties**

These properties are used to configure tRedshiftCommit running in the Standard Job framework.

The Standard tRedshiftCommit component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Component list</th>
<th>Select the tRedshiftConnection component in the list if more than one connection are planned for the current job.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Close Connection</th>
<th>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</th>
</tr>
</thead>
</table>

**Warning:**

*If you want to use a Row > Main connection to link tRedshiftCommit to your Job, your data will be committed row by row. In this case, do not select the Close Connection check box or your connection will be closed before the end of your first row commit.*

**Advanced settings**

<table>
<thead>
<tr>
<th>tStat Catcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other tRedshift* components, especially with the tRedshiftConnection and tRedshiftRollback components.</th>
</tr>
</thead>
</table>

| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

Related scenario

For a similar scenario, see Scenario: Inserting data in mother/daughter tables on page 2476.
tRedshiftConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

tRedshiftConnection opens a connection to the database for a current transaction.

**tRedshiftConnection Standard properties**

These properties are used to configure tRedshiftConnection running in the Standard Job framework. The Standard tRedshiftConnection component belongs to the Cloud and the Databases families. The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Host</td>
<td>Host name or IP address of DB server.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>Name of the schema.</td>
</tr>
</tbody>
</table>
| Username and Password | DB user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Additional JDBC Parameters | Specify additional JDBC properties for the connection you are creating. The properties are separated by ampersand & and each property is a key-value pair. For example, ssl=true & ssifactory=com.amazon.redshift.ssl.NonValidatingFactory, which means the connection will be created using SSL. |
Use or register a shared DB Connection

Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the Shared DB Connection Name field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the Use dynamic job and Use an independent process to run subjob options of the tRunJob component. Using a shared connection together with a tRunJob component with either of these two options enabled will cause your Job to fail.

Advanced settings

Auto Commit

Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared.

By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

tStatCatcher Statistics

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

Usage

Usage rule

This component is more commonly used with other tRedshift* components, especially with the tRedshiftCommit and tRedshiftRollback components.

Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).
Related scenario

For a related scenario, see Scenario: Handling data with Redshift on page 3139.
tRedshiftInput

Reads data from a database and extracts fields based on a query so that you may apply changes to the extracted data.

tRedshiftInput executes a DB query with a strictly defined order which must correspond to the schema definition. Then it passes on the field list to the next component through a Main row link.

### tRedshiftInput Standard properties

These properties are used to configure tRedshiftInput running in the Standard Job framework.

The Standard tRedshiftInput component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

#### Basic settings

<table>
<thead>
<tr>
<th>Property type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see **Talend Studio User Guide**.

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic**
settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Host</th>
<th>Hostname or IP address of the database server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of the database server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Database user authentication data. The properties are separated by ampersand &amp; and each property is a key-value pair. For example, <code>ssl=true &amp; ssifactory=com.amazon.redshift.ssl.NonValidatingFactory</code>, which means the connection will be created using SSL.</td>
</tr>
<tr>
<td>Additional JDBC Parameters</td>
<td>Specify additional JDBC properties for the connection you are creating. For example, <code>ssl=true &amp; ssifactory=com.amazon.redshift.ssl.NonValidatingFactory</code>, which means the connection will be created using SSL.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate</td>
</tr>
</tbody>
</table>
the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Name of the table from which the data will be read.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query Type and Query</strong></td>
<td>Enter your database query paying particularly attention to sequence the fields properly in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Guess Query</strong></td>
<td>Click the Guess Query button to generate the query which corresponds to your table schema in the Query field.</td>
</tr>
<tr>
<td><strong>Guess schema</strong></td>
<td>Click the Guess schema button to retrieve the table schema.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| Log file | Specify the path to the log file to be generated during the data retrieval process. Note that if the logging information needs to be saved and this component reuses the connection created by a tRedshiftConnection component, you must specify the property loglevel=1/2/3 in the Additional JDBC Parameters field of the tRedshiftConnection component. |
| Logging level | Select a logging level from the drop-down list to specify which kind of events will be logged in the log file. |
| Use cursor | Select this check box to help to decide the row set to work with at a time and thus optimize performance. |
| Trim all the String/Char columns | Select this check box to remove leading and trailing whitespace from all the String/Char columns. |
| Trim column | Remove leading and trailing whitespace from defined columns. |
| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer. **QUERY**: the query statement being processed. This is a Flow variable and it returns a string. **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible SQL queries for Amazon Redshift databases.</th>
</tr>
</thead>
</table>

**Dynamic settings**

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

**Scenario: Handling data with Redshift**

This scenario describes a Job that writes the personal information into Redshift, then retrieves the information in Redshift and displays it on the console.
The scenario requires the following six components:

- **tRedshiftConnection**: opens a connection to Redshift.
- **tFixedFlowInput**: defines the personal information data structure, and sends the data to the next component.
- **tRedshiftOutput**: writes the data it receives from the preceding component into Redshift.
- **tRedshiftInput**: reads the data from Redshift.
- **tLogRow**: displays the data it receives from the preceding component on the console.
- **tRedshiftClose**: closes the connection to Redshift.

**Dropping and linking the components**

**Procedure**

1. Drop the six components listed previously from the **Palette** onto the design workspace.
2. Connect **tFixedFlowInput** to **tRedshiftOutput** using a **Row > Main** connection.
3. Connect **tRedshiftInput** to **tLogRow** also using a **Row > Main** connection.
4. Connect **tRedshiftConnection** to **tFixedFlowInput** using a **Trigger > OnSubjobOk** connection.
5. Connect **tFixedFlowInput** to **tRedshiftInput** and **tRedshiftInput** to **tRedshiftClose** also using a **Trigger > OnSubjobOk** connection.
Configuring the components

Opening a connection to Redshift

Procedure

1. Double-click \texttt{tRedshiftConnection} to open its \textit{Basic settings} view.

   \begin{center}
   \includegraphics[width=\textwidth]{tRedshiftConnection_1}
   \end{center}

   \begin{itemize}
   \item Select \textit{Built-In} from the \textit{Property Type} drop-down list.
   \item In the \texttt{Host}, \texttt{Port}, \texttt{Database}, \texttt{Schema}, \texttt{Username}, and \texttt{Password} fields, enter the information required for the connection to Redshift.
   \item In \textit{Advanced settings} view, select \textit{Auto Commit} check box to commit any changes to Redshift upon each transaction.
   \end{itemize}

Defining the input data

Procedure

1. Double-click \texttt{tFixedFlowInput} to open its \textit{Basic settings} view.

   \begin{center}
   \includegraphics[width=\textwidth]{tFixedFlowInput_1}
   \end{center}

   \begin{itemize}
   \item Click the [...] button next to \textit{Edit schema} to open the schema editor.
   \end{itemize}
In the schema editor, click the [+ ] button to add three columns: id of the integer type, name of the string type, and age of the integer type.

Click OK to validate the changes and accept the propagation prompted by the pop-up [Propagate] dialog box.

In the Mode area, select Use Inline Content (delimited file) and enter the following personal information in the Content field.

Writing the data into Redshift

Procedure

1. Double-click tRedshiftOutput to open its Basic settings view.

2. Select the Use an existing connection check box, and then select the connection you have already configured for tRedshiftConnection from the Component List drop-down list.

3. In the Table field, enter or browse to the table into which you want to write the data, redshiftexample in this scenario.

4. Select Drop table if exists and create from the Action on table drop-down list, and select Insert from the Action on data drop-down list.

5. Click Sync columns to retrieve the schema from the preceding component tFixedFlowInput.
Retrieving the data from Redshift

Procedure

1. Double-click **tRedshiftInput** to open its **Basic settings** view.

   ![tRedshiftInput Basic settings view](image)

2. Select the **Use an existing connection** check box, and then select the connection you have already configured for **tRedshiftConnection** from the **Component List** drop-down list.

3. Click the [...] button next to **Edit schema** to open the schema editor.

4. In the schema editor, click the [+ ] button to add three columns: **id** of the integer type, **name** of the string type, and **age** of the integer type. The data structure is same as the structure you have defined for **tFixedFlowInput**.

5. Click **OK** to validate the changes and accept the propagation prompted by the pop-up **[Propagate]** dialog box.

6. In the **Table Name** field, enter or browse to the table into which you write the data, **redshiftexample** in this scenario.

7. Click the **Guess Query** button to generate the query. The **Query** field will be filled with the automatically generated query.

Displaying the defined information

Procedure

1. Double-click **tLogRow** to open its **Basic settings** view.

   ![tLogRow Basic settings view](image)

2. In the **Mode** area, select **Table (print values in cells of a table)** for a better view of the results.
Closing the connection to Redshift

Procedure

1. Double-click **tRedshiftClose** to open its **Basic settings** view.

   ![tRedshiftClose](image)

2. From **Component List**, select the connection you have already configured for **tRedshiftConnection**.

Saving and executing the Job

Procedure

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to execute the Job.

```
Starting job DOCTS151_tRedshift_S01 at 17:18 19/03/2014
[statistics] connecting to socket on port 3000
[statistics] connected
| tLogRow_1
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arthur</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>Ford</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>Jackson</td>
<td>17</td>
</tr>
</tbody>
</table>
[statistics] disconnected
Job DOCTS151_tRedshift_S01 ended at 17:18 19/03/2014. [exit code=0]
```

The personal information is written to the specified target Redshift database, and then the data is retrieved from the database and displayed on the console.
tRedshiftOutput

 Writes, updates, modifies or deletes the data in a database.

 tRedshiftOutput executes the action defined on the table and/or on the data of a table, according to the input flow from the previous component.

**tRedshiftOutput Standard properties**

These properties are used to configure tRedshiftOutput running in the Standard Job framework.

The Standard tRedshiftOutput component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository .</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>![Database icon]</td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.</td>
</tr>
<tr>
<td></td>
<td>For more information about setting up and storing database connection parameters, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>![Database icon]</td>
<td>Note: When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component</td>
</tr>
</tbody>
</table>
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Host</th>
<th>Hostname or IP address of the database server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of the database server.</td>
</tr>
<tr>
<td>Database</td>
<td>Database name.</td>
</tr>
<tr>
<td>Schema</td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Database user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Additional JDBC Parameters</td>
<td>Specify additional JDBC properties for the connection you are creating. The properties are separated by ampersand &amp; and each property is a key-value pair. For example, ssl=true &amp; sslfactory=com.amazon.redshift.ssl.NonValidatingFactory, which means the connection will be created using SSL.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to which the data will be written. Note that only one table can be written at a time.</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td><strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create a table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table if not exists</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop a table if exists and create</strong>: The table is removed if already exists and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear a table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td>Action on data</td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, the operation will stop.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Make changes to existing entries.</td>
</tr>
<tr>
<td></td>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
</tbody>
</table>
**Update or insert**: Update the record with the given reference. If the record does not exist, a new record would be inserted.

**Delete**: Remove entries corresponding to the input flow.

**Warning:**

*It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking Edit Schema and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the Advanced settings view where you can simultaneously define primary keys for the Update and Delete operations. To do that: Select the Use field options check box and then in the Key in update column select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the Key in delete column for the Delete operation.*

**Schema** and **Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

**Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.
### Die on error

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error through a Row > Rejects link.

### Advanced settings

#### Extend Insert

Select this check box to carry out a bulk insert of a defined set of lines instead of inserting lines one by one. The gain in system performance is considerable.

**Number of rows per insert:** enter the number of rows to be inserted per operation. Note that the higher the value specified, the lower performance levels shall be due to the increase in memory demands.

Amazon Redshift requires the number of rows per insert to be less than 32767. For this reason, if the number you enter exceeds this maximum limit, the Studio automatically resets this number below this limit.

**Note:**

This option is not compatible with the Reject link. You should therefore clear the check box if you are using a Row > Rejects link with this component.

#### Use Batch

Select this check box to activate the batch mode for data processing.

**Note:**

This check box is available only when you have selected the Update or the Delete option in the Action on data field.

#### Batch Size

Specify the number of records to be processed in each batch.

This field appears only when the Use batch mode check box is selected.

#### Commit every

Enter the number of rows to be completed before committing batches of rows together into the database. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.

#### Additional Columns

This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.

**Name:** Type in the name of the schema column to be altered or inserted as new column.

**SQL expression:** Type in the SQL statement to be executed in order to alter or insert the relevant column data.
**Position:** Select *Before*, *Replace* or *After* following the action to be performed on the reference column.

**Reference column:** Type in a column of reference that the tDBOutput can use to place or replace the new or altered column.

**Use field options**
Select this check box to customize a request, especially when there is double action on data.

**tStat Catcher Statistics**
Select this check box to collect log data at the component level.

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_LINE_UPDATED: the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_INSERTED: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_DELETED: the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_REJECTED: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <em>Ctrl + Space</em> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

### Usage

**Usage rule**
This component covers all possible SQL database queries. It allows you to carry out actions on a table or on the data of a table in an Amazon Redshift database. It enables you to create a reject flow, with a *Row > Rejects* link filtering the data in error. For a usage example, see *Scenario: Retrieving data in error with a Reject link* on page 2529.

**Dynamic settings**
Click the [*+*] button to add a row in the table and fill the *Code* field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful.
when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the <strong>Install</strong> button on the <strong>Component</strong> tab view. You can also find out and add all missing JARs easily on the <strong>Modules</strong> tab in the <strong>Integration</strong> perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

**Related scenarios**

For a related scenario, see **Scenario: Handling data with Redshift** on page 3139.
tRedshiftOutputBulk

Prepares a delimited/CSV file that can be used by tRedshiftBulkExec to feed Amazon Redshift.

The tRedshiftOutputBulk and tRedshiftBulkExec components can be used together in a two step process to load data to Amazon Redshift from a delimited/CSV file on Amazon S3. In the first step, a delimited/CSV file is generated. In the second step, this file is used in the INSERT statement used to feed Amazon Redshift. These two steps are fused together in the tRedshiftOutputBulkExec component. The advantage of using two separate steps is that the data can be transformed before it is loaded to Amazon Redshift.

This component receives data from the preceding component, generates a single delimited/CSV file and then uploads the file to Amazon S3.

tRedshiftOutputBulk Standard properties

These properties are used to configure tRedshiftOutputBulk running in the Standard Job framework.

The Standard tRedshiftOutputBulk component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data file path at local</td>
<td>Specify the local path to the file to be generated. Note that the file is generated on the same machine where the Studio is installed or where the Job using this component is deployed.</td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td>Built-In</td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Repository</td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Compress the data file**

Select this check box and select a compression type from the list displayed to compress the data file.

This check box disappears when the **Append the local file** check box is selected.

**Encrypt**

Select this check box to generate and upload the data file to Amazon S3 using client-side encryption. In the **Encryption key** field displayed, enter the encryption key used to encrypt the file.

By default, this check box is cleared and the data file will be uploaded to Amazon S3 using server-side encryption.

For more information about the client-side and server-side encryption, see Protecting Data Using Encryption.

**Access Key**

Specify the Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit Getting Your AWS Access Keys.

**Secret Key**

Specify the Secret Access Key, constituting the security credentials in combination with the access Key.

To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

**Bucket**

Type in the name of the Amazon S3 bucket, namely the top level folder, to which the file is uploaded.

**Key**

Type in an object key to assign to the file uploaded to Amazon S3.

**Advanced settings**

- **Field Separator**: Enter the character used to separate fields.
- **Text enclosure**: Select the character in a pair of which the fields are enclosed.
Delete local file after putting it to s3
Select this check box to delete the local file after being uploaded to Amazon S3. By default, this check box is selected.

Create directory if not exists
Select this check box to create the directory specified in the Data file path at local field if it does not exist. By default, this check box is selected.

Encoding
Select an encoding type for the data in the file to be generated.

Config client
Select this check box to configure client parameters for Amazon S3. Click the [+] button below the table displayed to add as many rows as needed, each row for a client parameter, and set the following attributes for each parameter:
- Client Parameter: Click the cell and select a parameter from the drop-down list.
- Value: Enter the value for the corresponding client parameter.

TStatCatcher Statistics
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Global Variables

**Global Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong>:</td>
<td>the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong>:</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
Related scenario

For a related scenario, see Scenario: loading/unloading data from/to Amazon S3 on page 3119.
tRedshiftOutputBulkExec

Executes the Insert action on the data provided.

As a dedicated component, it allows gains in performance during Insert operations to Amazon Redshift.

The tRedshiftOutputBulk and tRedshiftBulkExec components can be used together in a two step process to load data to Amazon Redshift from a delimited/CSV file on Amazon S3. In the first step, a delimited/CSV file is generated. In the second step, this file is used in the INSERT statement used to feed Amazon Redshift. These two steps are fused together in the tRedshiftOutputBulkExec component. The advantage of using two separate steps is that the data can be transformed before it is loaded to Amazon Redshift.

This component receives data from the preceding component, generates a single delimited/CSV file and uploads the file to Amazon S3, finally it loads the data from Amazon S3 to Redshift.

tRedshiftOutputBulkExec Standard properties

These properties are used to configure tRedshiftOutputBulkExec running in the Standard Job framework.

The Standard tRedshiftOutputBulkExec component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Either Built-In or Repository.</td>
</tr>
<tr>
<td>Host</td>
<td>Type in the IP address or hostname of the database server.</td>
</tr>
<tr>
<td>Port</td>
<td>Type in the listening port number of the database server.</td>
</tr>
</tbody>
</table>

Built-In: No property data stored centrally.

Repository: Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved.

Use an existing connection

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.
| **Database** | Type in the name of the database. |
| **Schema** | Type in the name of the schema. |
| **Username and Password** | Type in the database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Additional JDBC Parameters** | Specify additional JDBC properties for the connection you are creating. The properties are separated by ampersand & and each property is a key-value pair. For example, `ssl=true & sslfactory=com.amazon.redshift.ssl.NonValidatingFactory`, which means the connection will be created using SSL. |
| **Table Name** | Specify the name of the table to be written. Note that only one table can be written at a time. |
| **Action on table** | On the table defined, you can perform one of the following operations: <ul> • **None**: No operation is carried out. • **Drop and create table**: The table is removed and created again. • **Create table**: The table does not exist and gets created. • **Create table if not exists**: The table is created if it does not exist. • **Drop table if exists and create**: The table is removed if it already exists and created again. • **Clear table**: The table content is deleted. You have the possibility to rollback the operation. </ul> |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. <ul> • **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide. • **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. </ul> Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available: <ul> • **View schema**: choose this option to view the schema only. </ul>
• **Change to built-in property**: choose this option to change the schema to Built-in for local changes.

• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th><strong>Data file path at local</strong></th>
<th>Specify the local path to the file to be generated. Note that the file is generated on the same machine where the Studio is installed or where the Job using this component is deployed.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Append the local file</strong></td>
<td>Select this check box to append data to the specified local file if it already exists, instead of overwriting it.</td>
</tr>
<tr>
<td><strong>Create directory if not exists</strong></td>
<td>Select this check box to create the directory specified in the <strong>Data file path at local</strong> field if it does not exist. By default, this check box is selected.</td>
</tr>
<tr>
<td><strong>Access Key</strong></td>
<td>Specify the Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit <a href="#">Getting Your AWS Access Keys</a>.</td>
</tr>
<tr>
<td><strong>Secret Key</strong></td>
<td>Specify the Secret Access Key, constituting the security credentials in combination with the access Key. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Bucket</strong></td>
<td>Type in the name of the Amazon S3 bucket, namely the top level folder, to which the file is uploaded.</td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td>Type in an object key to assign to the file uploaded to Amazon S3.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Fields terminated by</strong></th>
<th>Enter the character used to separate fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enclosed by</strong></td>
<td>Select the character in a pair of which the fields are enclosed.</td>
</tr>
<tr>
<td><strong>Compressed by</strong></td>
<td>Select this check box and select a compression type from the list displayed to compress the data file. This field disappears when the <strong>Append the local file</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Encrypt</strong></td>
<td>Select this check box to generate and upload the data file to Amazon S3 using client-side encryption. In the <strong>Encryption key</strong> field displayed, specify the encryption key used to encrypt the file. Note that only a base64 encoded AES 128-bit or AES 256-bit envelope key is</td>
</tr>
</tbody>
</table>
supported. For more information, see [Loading Encrypted Data Files from Amazon S3](http://docs.aws.amazon.com/redshift/latest/dg/r_COPY.html).

By default, this check box is cleared and the data file will be uploaded to Amazon S3 using server-side encryption. For more information about the client-side and server-side encryption, see [Protecting Data Using Encryption](http://docs.aws.amazon.com/redshift/latest/dg/r_COPY.html).

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding</td>
<td>Select an encoding type for the data in the file to be generated.</td>
</tr>
<tr>
<td>Delete local file after putting it to S3</td>
<td>Select this check box to delete the local file after being uploaded to Amazon S3. By default, this check box is selected.</td>
</tr>
<tr>
<td>Date format</td>
<td>Select one of the following items from the list to specify the date format in the source data:</td>
</tr>
<tr>
<td></td>
<td>• <strong>NONE</strong>: No date format is specified.</td>
</tr>
<tr>
<td></td>
<td>• <strong>PATTERN</strong>: Select this item and specify the date format in the field displayed. The default date format is YYYY-MM-DD.</td>
</tr>
<tr>
<td></td>
<td>• <strong>AUTO</strong>: Select this item if you want Amazon Redshift to recognize and convert automatically the date format.</td>
</tr>
<tr>
<td>Time format</td>
<td>Select one of the following items from the list to specify the time format in the source data:</td>
</tr>
<tr>
<td></td>
<td>• <strong>NONE</strong>: No time format is specified.</td>
</tr>
<tr>
<td></td>
<td>• <strong>PATTERN</strong>: Select this item and specify the time format in the field displayed. The default time format is YYYY-MM-DD HH:MI:SS.</td>
</tr>
<tr>
<td></td>
<td>• <strong>AUTO</strong>: Select this item if you want Amazon Redshift to recognize and convert automatically the time format.</td>
</tr>
<tr>
<td></td>
<td>• <strong>EPOCHSECS</strong>: Select this item if the source data is represented as epoch time, the number of seconds since Jan 1, 1970 00:00:00 UTC.</td>
</tr>
<tr>
<td></td>
<td>• <strong>EPOCHMILLISECS</strong>: Select this item if the source data is represented as epoch time, the number of milliseconds since Jan 1, 1970 00:00:00 UTC.</td>
</tr>
<tr>
<td>Settings</td>
<td>Click the [+] button below the table to specify more parameters for loading the data.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Parameter</strong>: Click the cell and select a parameter from the drop-down list.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Value</strong>: Set the value for the corresponding parameter. Note that you cannot set the value for a parameter (such as <strong>IGNOREBLANKLINES</strong>)</td>
</tr>
</tbody>
</table>
### Config client

Select this check box to configure client parameters for Amazon S3. Click the [+] button below the table displayed to add as many rows as needed, each row for a client parameter, and set the following attributes for each parameter:

- **Client Parameter**: Click the cell and select a parameter from the drop-down list.
- **Value**: Enter the value for the corresponding client parameter.

### tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

#### Usage rule

This component is mainly used when no particular transformation is required on the data to be loaded to Amazon Redshift.

#### Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-*
Related scenario

For a related scenario, see Scenario: loading/unloading data from/to Amazon S3 on page 3119.
tRedshiftRollback

Cancels the transaction commit in the Redshift database to avoid committing part of a transaction involuntarily.

**tRedshiftRollback Standard properties**

These properties are used to configure tRedshiftRollback running in the Standard Job framework.
The Standard tRedshiftRollback component belongs to the Cloud and the Databases families.
The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tRedshiftConnection component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

### Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

### Usage

| Usage rule | This component is more commonly used with other tRedshift* components, especially with the tRedshiftConnection and tRedshiftCommit components. |
| Dynamic settings | Click the [+button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. |
Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Related scenario**

For a similar scenario, see **Scenario: Rollback from inserting data in mother/daughter tables** on page 2480.
tRedshiftRow

Acts on the actual DB structure or on the data (although without handling data), depending on the nature of the query and the database.

The SQLBuilder tool helps you write easily your SQL statements.

tRedshiftRow is the specific component for this database query. It executes the SQL query stated onto the specified database. The row suffix means the component implements a flow in the job design although it does not provide output.

**tRedshiftRow Standard properties**

These properties are used to configure tRedshiftRow running in the Standard Job framework.

The Standard tRedshiftRow component belongs to the Cloud and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>Note:</td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
<td></td>
</tr>
</tbody>
</table>
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](#).

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Hostname or IP address of the database server.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of the database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Exact name of the schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Additional JDBC Parameters</strong></td>
<td>Specify additional JDBC properties for the connection you are creating. The properties are separated by ampersand &amp; and each property is a key-value pair. For example, <code>ssl=true &amp; ssifactory=com.amazon.redshift.ssl.NonValidatingFactory</code>, which means the connection will be created using SSL.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
<tr>
<td><strong>Click Edit schema</strong></td>
<td>to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td>Table Name</td>
<td>Name of the table from which the data will be read.</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>Query type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: Fill in manually the query statement or build it graphically using SQLBuilder.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the relevant query stored in the Repository. The <strong>Query</strong> field gets accordingly filled in.</td>
</tr>
<tr>
<td>Guess Query</td>
<td>Click the <strong>Guess Query</strong> button to generate the query which corresponds to your table schema in the <strong>Query</strong> field.</td>
</tr>
<tr>
<td>Query</td>
<td>Enter your database query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error through a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Propagate QUERY’s recordset</th>
<th>Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the <strong>use column</strong> list.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of <strong>Object</strong> and this component is usually followed by <strong>tParseRecordSet</strong>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use PreparedStatement</th>
<th>Select this check box if you want to query the database using a PreparedStatement. In the <strong>Set PreparedStatement Parameter</strong> table, define the parameters represented by &quot;?&quot; in the SQL instruction of the <strong>Query</strong> field in the <strong>Basic Settings</strong> tab.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter Index</td>
<td>Enter the parameter position in the SQL instruction.</td>
</tr>
<tr>
<td>Parameter Type</td>
<td>Enter the parameter type.</td>
</tr>
<tr>
<td>Parameter Value</td>
<td>Enter the parameter value.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>This option is very useful if you need to execute the same query several times. Performance levels are increased.</td>
</tr>
</tbody>
</table>

| Commit every | Number of rows to be completed before committing batches of rows together into the database. This option |
ensures transaction quality (but not rollback) and above all better performance on executions.

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

### Global Variables

**Global Variables**

<table>
<thead>
<tr>
<th>QUERY</th>
<th>the query statement being processed. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

**Usage rule**

This component offers the flexibility benefit of the database query and covers all possible SQL queries.

**Dynamic settings**

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided.
Related scenarios

For similar scenarios, see:

- Scenario: Combining two flows for selective output on page 2558
- Procedure on page 625
- Scenario: Removing and regenerating a MySQL table index on page 2552
tRedshiftUnload

Unloads data on Amazon Redshift to files on Amazon S3.
This component runs a specified query in Amazon Redshift and then unloads the result of the query to
one or more files on Amazon S3.

**tRedshiftUnload Standard properties**

These properties are used to configure tRedshiftUnload running in the Standard Job framework.
The Standard tRedshiftUnload component belongs to the Cloud and the Databases families.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-In:</strong> No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**Host**

Type in the IP address or hostname of the database server.

**Port**

Type in the listening port number of the database server.

**Database**

Type in the name of the database.

**Schema**

Type in the name of the schema.

**Username and Password**

Type in the database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**Additional JDBC Parameters**

Specify additional JDBC properties for the connection you are creating. The properties are separated by ampersand & and each property is a key-value pair. For example, `ssl=true & ssifactory=com.amazon.redshift.ssl.NonValidatingFactory`, which means the connection will be created using SSL.

**Table Name**

Type in the name of the table from which the data will be read.
### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

**Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](#).

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](#).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Query Type and Query

Enter the database query paying particularly attention to the proper sequence of the fields in order to match the schema definition.

### Guess Query

Click the button to generate the query which corresponds to the table schema in the **Query** field.

### Access Key

Specify the Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit [Getting Your AWS Access Keys](#).

### Secret Key

Specify the Secret Access Key, constituting the security credentials in combination with the access Key.

To enter the secret key, click the `[...]` button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

### Bucket

Type in the name of the Amazon S3 bucket, namely the top level folder, to which the data is unloaded.

### Key prefix

Type in the name prefix for the unload files on Amazon S3. By default, the unload files are written per slice of the Redshift cluster and the file names are written in the format `<object_path>/<name_prefix><slice-number>_part_<file-number>`.
### Advanced settings

| **File type** | Select the type of the unload files on Amazon S3 from the list:  
| | • Delimited file or CSV: a delimited/CSV file.  
| | • Fixed width: a fixed-width file. |

| **Fields terminated by** | Enter the character used to separate fields.  
| | This field appears only when Delimited file or CSV is selected from the File type list.  

| **Enclosed by** | Select the character in a pair of which the fields are enclosed.  
| | This list appears only when Delimited file or CSV is selected from the File type list.  

| **Fixed width mapping** | Enter a string that specifies a user-defined column label and column width between double quotation marks. The format of the string is:  
| | ColumnLabel1:ColumnWidth1,ColumnLabel2:ColumnWidth2,...  
| | Note that the column label in the string has no relation to the table column name and it can be either a text string or an integer. The order of the label/width pairs must match the order of the table columns exactly.  
| | This field appears only when Fixed width is selected from the File type list. |

| **Compressed by** | Select this check box and from the list displayed select the compression type of the files. |

| **Encrypt** | Select this check box to encrypt unload file(s) using Amazon S3 client-side encryption. In the Encryption key field displayed, enter the encryption key used to encrypt the unload file(s). Note that only a base64 encoded AES 128-bit or AES 256-bit envelope key is supported. For more information, see Unloading Encrypted Data Files. |

| **Specify null string** | Select this check box and from the list displayed select a string that represents a null value in unload files. |

| **Escape** | Select this check box to place an escape character (\) before every occurrence of the following characters for CHAR and VARCHAR columns in the delimited unload files: linefeed (\n), carriage return (\r), the delimiter character specified for the unloaded data, the escape character (\), a quote character (" or "). |

| **Overwrite s3 object if exist** | Select this check box to overwrite the existing Amazon S3 object files. |

| **Parallel** | Select this check box to write data in parallel to multiple unload files on Amazon S3 according to the number of slices in the Redshift cluster. |
tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Global Variables

Global Variables

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**

This component covers all possible SQL queries for the Amazon Redshift database.

**Dynamic settings**

Click the [*] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

Related Scenario

For a related scenario, see *Scenario: loading/unloading data from/to Amazon S3* on page 3119.
tReplace

Cleanses all files before further processing.
Carries out a Search & Replace operation in the input columns defined.

tReplace Standard properties

These properties are used to configure tReplace running in the Standard Job framework.
The Standard tReplace component belongs to the Processing family.
The component in this framework is generally available.

Basic settings

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:
• View schema: choose this option to view the schema only.
• Change to built-in property: choose this option to change the schema to Built-in for local changes.
• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.
Two read-only columns, Value and Match are added to the output schema automatically. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-In</td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Repository</td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
| Simple Mode Search / Replace | Click the button to add as many conditions as needed. The conditions are performed one after the other for each row.
Input column: Select the column of the schema the search & replace is to be operated on.
Search: Type in the value to search in the input column. |
| **Replace with**: Type in the substitution value. |
| **Whole word**: Select this check box if the searched value is to be considered as whole. |
| **Case sensitive**: Select this check box to care about the case. |

Note that you cannot use regular expression in these columns.

### Use advanced mode

Select this check box when the operation you want to perform cannot be carried out through the simple mode. In the text field, type in the regular expression as required.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the job processing metadata at a job level as well as at each component level.

### Global Variables

**Global Variables**

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.
- **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

**Usage rule**

This component is not startable as it requires an input flow. And it requires an output component.

### Scenario 1: Multiple replacements and column filtering

This following Job searches and replaces various typos and defects in a csv file then operates a column filtering before producing a new csv file with the final output.
• Drop the following components from the Palette onto the design workspace: tFileInputDelimited, tReplace, tFilterColumn and tFileOutputDelimited.

• Connect the components using Main Row connections via a right-click each component.

• Select the tFileInputDelimited component and set the input flow parameters.

  Property Type: Built-In
  File name: 'D:\Input\replace.csv'
  Row Separator: "\r"  Field Separator: ","  Header: 0  Footer: 0
  Schema: Built-In  Edit schema: ...
  Extract lines at random
  Encoding Type: ISO-8859-15

• The File is a simple csv file stored locally. The Row Separator is a carriage return and the Field Separator is a semi-colon. In the Header is the name of the column, and no Footer nor Limit are to be set.

• The file contains characters such as: *t, . or Nikson which we want to turn into Nixon, and streat, which we want to turn into Street.

<table>
<thead>
<tr>
<th>Street</th>
<th>FirstName</th>
<th>Name</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>street</td>
<td>John</td>
<td>Kennedy</td>
<td>98.30$</td>
</tr>
<tr>
<td>street</td>
<td>Richad</td>
<td>Nikson</td>
<td>78.23$</td>
</tr>
<tr>
<td>street</td>
<td>Richard</td>
<td>Nikson</td>
<td>78.2$</td>
</tr>
<tr>
<td>street</td>
<td>toto</td>
<td>Nikson</td>
<td>78.23$</td>
</tr>
<tr>
<td>street</td>
<td>Richard</td>
<td>Nikson</td>
<td>78.23$</td>
</tr>
<tr>
<td>street</td>
<td>Georges</td>
<td>bush</td>
<td>99.99$</td>
</tr>
</tbody>
</table>

• The schema for this file is built in also and made of four columns of various types (string or int).

• Now select the tReplace component to set the search & replace parameters.
- The schema can be synchronized with the incoming flow.

- Select the **Simple mode** check box as the search parameters can be easily set without requiring the use of regexp.

- Click the plus sign to add some lines to the parameters table.

- On the first parameter line, select **Amount** as **InputColumn**. Type "." in the **Search** field, and "," in the **Replace** field.

- On the second parameter line, select **Street** as **InputColumn**. Type "street" in the **Search** field, and "Street" in the **Replace** field.

- On the third parameter line, select again **Amount** as **InputColumn**. Type "$" in the **Search** field, and "£" in the **Replace** field.

- On the fourth parameter line, select **Name** as **InputColumn**. Type "Nikson" in the **Search** field, and "Nixon" in the **Replace** field.

- On the fifth parameter line, select **Firstname** as **InputColumn**. Type "*t" in the **Search** field, and replace them with nothing between double quotes.

- The advanced mode isn’t used in this scenario.

- Select the next component in the Job, **tFilterColumn**.

- The **tFilterColumn** component holds a schema editor allowing to build the output schema based on the column names of the input schema. In this use case, add one new column named **empty_field**
and change the order of the input schema columns to obtain a schema as follows: empty field, Firstname, Name, Street, Amount.

- Click OK to validate.

- Set the tFileOutputDelimited properties manually.

- The schema is built-in for this scenario, and comes from the preceding component in the Job.

- Save the Job and press F6 to execute it.

<table>
<thead>
<tr>
<th>John</th>
<th>Kennedy</th>
<th>Street</th>
<th>98,30£</th>
</tr>
</thead>
<tbody>
<tr>
<td>Richad</td>
<td>Nixon</td>
<td>Street</td>
<td>78,23£</td>
</tr>
<tr>
<td>Richard</td>
<td>Nixon</td>
<td>Street</td>
<td>78,2£</td>
</tr>
<tr>
<td>toto</td>
<td>Nixon</td>
<td>Street</td>
<td>78,23£</td>
</tr>
<tr>
<td>Richard</td>
<td>Nixon</td>
<td>Street</td>
<td>78,23£</td>
</tr>
<tr>
<td>Georges</td>
<td>bush</td>
<td>street</td>
<td>99,99£</td>
</tr>
</tbody>
</table>

The first column is empty, the rest of the columns have been cleaned up from the parasitical characters, and Nikson was replaced with Nixon. The street column was moved and the decimal delimiter has been changed from a dot to a comma, along with the currency sign.
tReplaceList

Cleanses all files before further processing.
Carries out a Search and Replace operation in the input columns defined based on an external lookup.

tReplaceList Standard properties

These properties are used to configure tReplaceList running in the Standard Job framework.
The Standard tReplaceList component belongs to the Data Quality family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository. Two read-only columns, Value and Match are added to the output schema automatically.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Warning:</strong> The data Type defined in the schemas must be consistent, ie., an integer can only be replaced by another integer using an integer as a look up field. Values of one type cannot be replaced by values of another type.</td>
</tr>
<tr>
<td>Built-in:</td>
<td>The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Repository:</td>
<td>The schema already exists and is stored in the Repository, hence can be reused in various projects and job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Lookup search column

Select the column to be searched in the lookup schema.

Lookup replacement column

Select the column where the replacement values are stored.

Column options

Select the columns of the main flow where the replacement is to be carried out.

Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |
Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | tReplaceList is an intermediary component. It requires an input flow and an output component. |

Scenario: Replacement from a reference file

The following Job searches and replaces a list of states with their corresponding two-letter codes. The relevant codes are taken from a reference file placed as lookup flow in the Job.

Setting up the Job

Procedure

1. Drop the following components from the Palette onto the design workspace: two tFileInputDelimited components, a tReplaceList and a tLogRow.
2. Connect the two tFileInputDelimited components to the tReplaceList component using Row > Main connections. Note that the link between the reference input component (the second tFileInputDelimited) and the tReplaceList component appears as a lookup row.
3. Connect the tReplaceList component to the tLogRow component using a Row > Main connection.
Configuring the components

Procedure

1. Double-click the first tFileInputDelimited component to open its Basic settings view and set the parameters of the main input flow, including the path and name of the file to read and the number of header rows to skip.

![tFileInputDelimited_1](image)

In this example, the main input file provides a list of people names and US state names. The following shows an extract of the file content:

```
name;state
Andrew Kennedy;Mississippi
Benjamin Carter;Louisiana
Benjamin Monroe;West Virginia
Bill Harrison;Tennessee
Calvin Grant;Virginia
Chester Harrison;Rhode Island
Chester Hoover;Kansas
Chester Kennedy;Maryland
Chester Polk;Indiana
Dwight Nixon;Nevada
Dwight Roosevelt;Mississippi
Franklin Grant;Nebraska
```

2. Click the [...] button next to Edit schema to open the [Schema] dialog box and set the input schema.

According to the structure of the main input file, the input schema should contain two columns: `name` and `state`. 

When done, click OK to close the dialog box and propagate the changes to the next component.

3. Define the properties of the second tFileInputDelimited component similarly.

In this example, the reference input file provides a list of states and their two-letter codes. Accordingly, the reference input schema should have two columns: state and code.

4. Double-click the tReplaceList component to open its Basic settings view to set the operation to carry out.
5. From the **Lookup search column** list, select the column to be searched. In this use case, we want to carry out a search on the *state* column.

6. From the **Lookup replacement column** list, select the column containing the replacement values, *code* for the two-letter state codes in this example.

7. In the **Column options** table, select **Replace** check box for the *states* column, to replace the state names with their corresponding codes.

8. In the **tLogRow** component, select the **Table** check box for a better readability of the output.

### Executing the Job

#### Procedure

Save the Job and press **F6** to execute it.

The state names have been replaced with their respective two-letter codes.
tReplicate

Duplicates the incoming schema into two identical output flows.
This component performs different operations on the same schema.

tReplicate Standard properties

These properties are used to configure tReplicate running in the Standard Job framework.
The Standard tReplicate component belongs to the Orchestration family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. Click Sync columns to retrieve the schema from the previous component in the Job.</td>
<td></td>
</tr>
<tr>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>

Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions</th>
</tr>
</thead>
</table>
Replicate 3183 only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

---

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is not startable (green background), it requires an Input component and an output component.</th>
</tr>
</thead>
</table>

**Connections**

Outgoing links (from this component to another):
- **Row**: Main.
- **Trigger**: Run if; On Component Ok; On Component Error.

Incoming links (from one component to this one):
- **Row**: Main; Reject;

For further information regarding connections, see *Talend Studio User Guide*.

---

**Scenario: Replicating a flow and sorting two identical flows respectively**

The scenario describes a Job that reads an input flow which contains names and states from a CSV file, replicates the input flow, then sorts the two identical flows based on name and state respectively, and displays the sorted data on the console.
**Setting up the Job**

**Procedure**

1. Drop the following components from the **Palette** to the design workspace: one **tFileInputDelimited** component, one **tReplicate** component, two **tSortRow** components, and two **tLogRow** components.
2. Connect **tFileInputDelimited** to **tReplicate** using a **Row > Main** link.
3. Repeat the step above to connect **tReplicate** to two **tSortRow** components respectively and connect **tSortRow** to **tLogRow**.
4. Label the components to better identify their functions.

**Configuring the components**

**Procedure**

1. Double-click the **tFileInputDelimited** component to open its **Basic settings** view in the **Component** tab.

   ![Source(tFileInputDelimited_1)](image)

   2. Click the [...] button next to the **File name/Stream** field to browse to the file from which you want to read the input flow. In this example, the input file is *Names&States.csv*, which contains two columns: *name* and *state*.

   ```
   name;state
   Andrew Kennedy;Mississippi
   Benjamin Carter;Louisiana
   Benjamin Monroe;West Virginia
   Bill Harrison;Tennessee
   Calvin Grant;Virginia
   Chester Harrison;Rhode Island
   Chester Hoover;Kansas
   Chester Kennedy;Maryland
   Chester Polk;Indiana
   Dwight Nixon;Nevada
   Dwight Roosevelt;Mississippi
   Franklin Grant;Nebraska
   ```

   3. Fill in the **Header**, **Footer** and **Limit** fields according to your needs. In this example, type in 1 in the **Header** field to skip the first row of the input file.
4. Click **Edit schema** to define the data structure of the input flow.

![Schema of Source](image)

5. Double-click the first **tSortRow** component to open its **Basic settings** view.

![Sort by name(tSortRow_1)](image)

6. In the **Criteria** panel, click the [+](image) button to add one row and set the sorting parameters for the schema column to be processed. To sort the input data by name, select **name** under **Schema column**. Select **alpha** as the sorting type and **asc** as the sorting order.

For more information about those parameters, see **tSortRow Standard properties** on page 3577.

7. Double-click the second **tSortRow** component and repeat the step above to define the sorting parameters for the state column.

![Sort by state(tSortRow_2)](image)
8. In the Basic settings view of each tLogRow component, select Table in the Mode area for a better view of the Job execution result.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Execute the Job by pressing **F6** or clicking **Run** on the Run tab.

The data sorted by name and state are both displayed on the console.
**tREST**

Serves as a REST Web service client.

The tREST component sends HTTP requests to a REpresentational State Transfer (REST) Web service provider and gets responses correspondingly.

**tREST Standard properties**

The Standard tREST component belongs to the Internet family.

**Note:**

To build Jobs that need to be deployed into the Talend Runtime, it is recommended to use the tRESTClient component which is best suited for the Talend Runtime.

For more information about the tRESTClient component, see tRESTClient on page 3191.

The component in this framework is generally available.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. This component always uses a built-in, read-only schema that contains two columns:  
- **Body**: stores the result from the server end.  
- **ERROR_CODE**: stores the HTTP status code from the server end when an error occurs during the invocation process. The specific meanings of the errors codes are subject to definitions of your Web service provider. For reference information, visit en.wikipedia.org/wiki/List_of_HTTP_status_codes.  
Click Edit Schema to view the schema structure.  
**Warning:** Changing the schema type may result in loss of the schema structure and therefore failure of the component. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>URL</strong></td>
<td>Type in the URL address of the REST Web server to be invoked.</td>
</tr>
</tbody>
</table>
| **HTTP Method** | From this list, select an HTTP method that describes the desired action. The specific meanings of the HTTP methods are subject to definitions of your Web service provider. Listed below are the generally accepted HTTP method definitions:  
- **GET**: retrieves data from the server end based on the given parameters.  
- **POST**: creates and uploads data based on the given parameters. |
| **- PUT**: updates data based on the given parameters, or if the data does not exist, creates it.  
**- DELETE**: removes data based on the given parameters. |
| --- |
| **HTTP Headers**  
Type in the name-value pair(s) for HTTP headers to define the parameters of the requested HTTP operation.  
For the specific definitions of HTTP headers, consult your REST Web service provider. For reference information, visit en.wikipedia.org/wiki/List_of_HTTP_headers. |
| **HTTP Body**  
Type in the payload to be uploaded to the server end when the **POST** or **PUT** action is selected. |
| **Advanced settings** |
| **tStatCatcher Statistics**  
Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |
| **Global Variables** |
| **Global Variables**  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see **Talend Studio User Guide**. |
| **Usage** |
| **Usage rule**  
Use this component as a REST Web service client to communicate with a REST Web service provider. It must be linked to an output component. |
| **Limitation**  
JRE 1.6 must be running for this component to work properly. |

**Scenario: Creating and retrieving data by invoking REST Web service**

This scenario describes a simple Job that invokes a REST Web service to create a new customer record on the server end and then retrieve the customer information. When executed, the Job displays relevant information on the **Run** console.
• Drop the following components from the Palette onto the design workspace: two tREST components and two tLogRow components, and label the two tREST components to best describe the actions to perform.

• Connect each tREST to one tLogRow using a Row > Main connection.

• Connect the first tREST to the second tREST using a Trigger > OnSubjobOK connection.

• Double click the first tREST component to open its Basic settings view.

  ![Create_customer(tREST_1)](image)

  - Fill the URL field with the URL of the Web service you are going to invoke. Note that the URL provided in this use case is for demonstration purposes only and is not a live address.
  - From the HTTP Method list, select POST to send an HTTP request for creating a new record.
  - Click the plus button to add a line in the HTTP Headers table, and type in the appropriate name-value key pair, which is subject to definition of your service provider, to indicate the media type of the payload to send to the server end. In this use case, type in Content-Type and application/xml. For reference information about Internet media types, visit www.w3.org/Protocols/rfc2616/rfc2616-sec3.html#sec3.7.
  - Fill the HTTP Body field with the payload to be uploaded to the server end. In this use case, type in <Customer><name>Steven</name></Customer> to create a record for a new customer named Steven.

  **Note:**
If you want to include double quotation marks in your payload, be sure to use a backslash escape character before each of the quotation marks. In this use case, for example, type in `<Customer><name>"Steven"</name></Customer>` if you want to enclose the name Steven in a pair of double quotation marks.

• Double click the second tREST component to open its Basic settings view.

• Fill the URL field with the same URL.

• From the HTTP Method list, select GET to send an HTTP request for retrieving the existing records.

• In the Basic settings view of each tLogRow, select the Print component unique name in front of each output row and Print schema column name in front of each value check boxes for better identification of the output flows.

• Save your Job and press F6 to launch it.

The console shows that the first tREST component sends an HTTP request to the server end to create a new customer named Steven, and the second tREST component successfully reads data from the server end, which includes the information of the new customer you just created.
tRESTClient

Interacts with RESTful Web service providers by sending HTTP and HTTPS requests using CXF (JAX-RS) getting the corresponding responses.

This component integrates well with Talend Runtime to get HTTPS support, with more QoS features to be supported in time.

**tRESTClient Standard properties**

These properties are used to configure tRESTClient running in the Standard Job framework.

The Standard tRESTClient component belongs to the ESB family.

The component in this framework is generally available.

**Basic settings**

| URL | Type in the URL address of the REST server to be invoked. When the Use Service Locator check box is selected, this field will not show and the URL of the REST server will be obtained from the Service Locator server automatically. |
| Relative Path | Enter the relative path of the REST server to be invoked. For example, if you want to access http://localhost:8888/services/Customers/list: If Use Service Locator is disabled: You can enter any of the first part of the address in the URL field, and the second part in the Relative Path field. For example, you can enter http://localhost:8888 in URL and /services/Customers/list in Relative Path. You can also enter the full path of the REST server in URL and leave Relative Path blank. If Use Service Locator is enabled: The URL part will be given by the Service Locator. In this case, you need to know the URL part, and specify the rest part in Relative Path. This depends on the service you request. For example, on tRESTRequest, you specify REST Endpoint as http://localhost:8888/services and enable Use Service Locator. Then, if you want to use this service, on tRESTClient side, you should specify /customers/list in Relative Path. |
| HTTP Method | From this list, select an HTTP method that describes the desired action. The specific meanings of the HTTP methods are subject to definitions of your Web service provider. Listed below are the generally accepted HTTP method definitions:  
- **GET**: retrieves data from the server end based on the given parameters.  
- **POST**: uploads data to the server end based on the given parameters. |
#### tRESTClient

| PUT: updates data based on the given parameters, or if the data does not exist, creates it. |
| PATCH: modifies data partially based on the given parameters. |
| DELETE: removes data based on the given parameters. |

### Content Type

Select **XML**, **JSON**, or **FORM** according to the media type of the content to be uploaded to the server end. This list appears only when you select the **POST**, **PUT**, or **PATCH** HTTP method.

### Accept Type

Select the media type the client end is prepared to accept for the response from the server end. Available options are **XML**, **JSON**, and **ANY**. When **ANY** is selected, the response message can be of any type and will be transformed into a string.

### Query parameters

Specify the URI query parameters in the form of name-value pairs. This option is mostly used with the GET method.

### Schema and Edit Schema

A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. This component uses three built-in, read-only schemas. Click **Edit Schema** to view the schema structure.

**Warning:**

*Changing the schema type may result in loss of the schema structure and therefore failure of the component.*

### Input Schema

Schema for the input data. This schema contains two columns:
- **body**: stores the content of structured input data
- **string**: stores the input content when it is, or is handled as, a string.

### Response Schema

Schema for server response. This schema is passed onto the next component via a **Row > Response** link, and it contains three columns:
- **statusCode**: stores the HTTP status code from the server end.
- **body**: stores the content of a structured response from the server end.
- **string**: stores the response content from the server end when it is, or is handled as, a string.

### Error Schema

Schema for error information. This schema is passed onto the next component via a **Row > Error** link, and it contains two columns:
- **errorCode**: stores the HTTP status code from the server end when an error occurs during the invocation
process. The specific meanings of the errors codes are subject to definitions of your Web service provider. For reference information, visit en.wikipedia.org/wiki/List_of_HTTP_status_codes.

- **errorMessage**: stores the error message corresponding the error code.

<table>
<thead>
<tr>
<th>Use Service Locator</th>
<th>Select this check box to enable the Service Locator. It maintains the availability of the service to help meet demands and service level agreements (SLAs). Specify the Service namespace and the Service name in the corresponding fields.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Use Service Activity Monitor</th>
<th>Select this check box to enable the Service Activity Monitor. It captures events and stores this information to facilitate in-depth analysis of service activity and track-and-trace of messages throughout a business transaction. This can be used to analyze service response times, identify traffic patterns, perform root cause analysis and more.</th>
</tr>
</thead>
</table>

| Use Authentication | Select this check box if authentication is required on the REST server end. Select the authentication type from:

  - **Basic HTTP**: Fill the **Username** and **Password** fields with your credentials.
  - **HTTP Digest**: Fill the **Username** and **Password** fields with your credentials.
  - **SAML Token (ESB runtime only)**: Fill the **Username** and **Password** fields with your credentials.
  - **OAuth2 Bearer**: Fill the **Bearer Token** field with a base64-encoded credential string.

  To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

| Use Business Correlation | Select this check box to create a correlation ID in this component.

  You can specify a correlation ID in the **Correlation Value** field. In this case the correlation ID will be passed on to the service it calls so that chained service calls will be grouped under this correlation ID. If you leave this field empty, this value will be generated automatically at runtime.

  When this option is enabled, **tRESTClient** will also extract the correlation ID from the response header and store it in the component variable for further use in the flow. |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

<p>| Die on error | This check box is selected to kill the Job when an error occurs. Clear the check box to skip the row on error and complete the process for error-free rows. |</p>
<table>
<thead>
<tr>
<th>Advanced settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Log messages</strong></td>
<td>Select this check box to log the message exchange between the service provider and the consumer.</td>
</tr>
<tr>
<td><strong>Enable WebClient Operation Reporting</strong></td>
<td>If this check box is selected, a CXF endpoint property will be set on the client side which may be used by some outgoing CXF metrics interceptors, to track requests to services using their URL's. The property set is <code>org.apache.cxf.resource.operation.name</code> and it contains service URL together with the HTTP method name, for example <code>GET:http://example.com:8080/service/</code>. If this check box is not selected, the property will not be set and some metrics will not be counted, which may increase the performance.</td>
</tr>
<tr>
<td><strong>Convert Response To DOM Document</strong></td>
<td>Select this check box to convert the response from the server to document type. Clear this check box if you want the response to be handled as a string.</td>
</tr>
<tr>
<td><strong>Convert Types To Strings</strong></td>
<td>This option appears when Content Type is JSON, or Accept Type is JSON or any. Select this check box to convert all values in JSON to string between quotation marks. For example, with this option enabled, the JSON data:</td>
</tr>
</tbody>
</table>
| | ```json
"root": {
  "test": 111
}
``` |
| | will be changed to: |
| | ```json
"root": {
  "test": "111"
}
``` |
| **Drop JSON Request Root** | This option appears when HTTP Method is POST, PUT or PATCH and Content Type is JSON. Select this check box to drop root JSON elements. |
| **Wrap JSON Response** | This option appears and is enabled by default when JSON is selected from the Accept Type list in the Basic settings view. With this check box selected, the response is wrapped with a root element. Clear this check box to remove the root element from the response. |
| **HTTP Headers** | Type in the name-value pair(s) for HTTP headers to define the parameters of the requested HTTP operation. For the specific definitions of HTTP headers, consult your REST Web service provider. For reference |
### Disable chunked encoding
This option appears when **HTTP Method** is **POST**, **PUT** or **PATCH**. Select this check box to disable encoding the payload as chunks.

### Service Locator Customer Properties
This option appears when **Use Service Locator** is enabled in the **Basic settings** tab. Click [+] to add as many properties as needed to the table. Enter the name and the value of each property in the **Property Name** field and the **Property Value** field respectively to identify the service.

### Service Activity Customer Properties
This option appears when **Use Service Activity Monitor** is enabled in the **Basic settings** tab. Click [+] to add as many properties as needed to the table. Enter the name and the value of each property in the **Property Name** field and the **Property Value** field respectively to identify the service.

### Connection timeout
Set the amount of time, in seconds, that the client will attempt to establish a connection before it times out. If set to 0, the client will continue to attempt to open a connection indefinitely. (default: 30)

This option only works in the Studio. To use it after the component is deployed in runtime, in the configuration file <TalendRuntimePath>/container/etc/org.apache.cxf.http.conduits-common.cfg, add `url = http://.*` to handle both HTTP and HTTPS requests, and specify the `client.ConnectionTimeout` parameter in milliseconds. If you need to use the **Receive timeout** option, specify the `client.ReceiveTimeout` in milliseconds too.

### Receive timeout
Set the amount of time, in seconds, that the client will wait for a response before it times out. If set to 0, the client will wait indefinitely. (default: 60) This option only works in the studio. For how to use it after the component is deployed in runtime or in Talend Administration Center, see the **Connection timeout** option.

### Use HTTP proxy
Select this check box if you are using a proxy server. Once selected, you need to provide the connection details: host, port, username and password.

### tStatCatcher Statistics
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

| Global Variables | NB_LINE: the number of rows processed. This is an After variable and it returns an integer. |
**tRESTClient**

**HEADERS**: the HTTP response headers. This is a Flow variable and it returns a list of HTTP response header values.

**CORRELATION_ID**: the correlation ID by which chained service calls will be grouped. This is a Flow variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.

**Usage**

**Usage rule**

This component is used as a RESTful Web service client to communicate with a RESTful service provider, with the ability to input a request to a service into a Job and return the Job result as a service response. Depending on the actions to perform, it usually works as a start or middle component in a Job or subjob.

**Dynamic settings**

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to turn on or off the **Use Authentication** or **Use HTTP proxy** option dynamically at runtime. You can add two rows in the table to set both options.

Once a dynamic parameter is defined, the corresponding option becomes highlighted and unusable in the **Basic settings** view or **Advanced settings** view.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Connections**

**Outgoing links:**

- **Row**: Response; Error.

**Trigger**: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error.

**Incoming links:**

- **Row**: Main; Reject.
Scenario 1: Getting user information by interacting with a RESTful service

This scenario describes a three-component Job that retrieves user information based on the user ID from a REST service via HTTP GET and displays the retrieved user information, as well as the message exchange between the client and the server, on the Run console.

Prerequisites:

If you are a Talend Open Studio for ESB user, create a Job as described in Scenario 2: Using URI Query parameters to explore the data of a database on page 3219, run the Job to expose a REST service, and enter the REST service URL in your Web browser, http://localhost:8088/users in this example. You should see information like the following:

```xml
- <users>
  - <user id="1">
    <first_name>Ronald</first_name>
    <last_name>Johnson</last_name>
  </user>
  - <user id="2">
    <first_name>Theodore</first_name>
    <last_name>Harding</last_name>
  </user>
  - <user id="3">
    <first_name>Bill</first_name>
    <last_name>Roosevelt</last_name>
  </user>
</users>
```

If you are not a Talend Open Studio for ESB user, then you need to get from your REST service provider the URL, the data structure, and the required parameters of the REST service you are going to call and make necessary modifications in the scenario configurations accordingly.

Setting up the Job

Procedure

1. Drop the following components from the Palette onto the design workspace:
   - tRESTClient, used to call the REST service and retrieve user information from the server end,
   - tXMLMap, used to adapt the tree structure of the REST service, and
- tLogRow, to display the retrieved user information on the Run console.

2. Connect the tRESTClient to the tXMLMap using a Row > Response connection.

3. Connect the tXMLMap to the tLogRow using a Row > Main connection, and give it a name, out in this example.

4. Label the components to best describe the actions to perform.

Configuring the components

Configuring the service call

Procedure

1. Double-click the tRESTClient component to open its Basic settings view.

2. Fill the URL field with the URL of the REST service you are going to invoke, "http://localhost:8088/users" in this example. Note that the URL provided in this use case is for demonstration purposes only and not a live address.

3. From the HTTP Method list, select GET to send an HTTP request for retrieving the existing records.
From the **Accept Type** list, select the type the client end is prepared to accept for the response from the server end, **XML**. Leave the rest of the settings as they are.

4. Click the [+] button beneath the **Query parameters** table to add two parameters, *from* and *to*, and set both parameters to 2, to get the information of the user with the ID of 2.

   Alternatively, you can query the information of the user with the ID of 2 by adding `?from=2&to=2` to the service URL.

5. In the **Advanced settings** view of the **tRESTClient** component, select the **Log messages** and the **Convert Response To DOM Document** check boxes to log the message exchange to the server and convert the response from the server to document type.

**Mapping the service structure and displaying the retrieved user information**

**Procedure**

1. Double-click the **tXMLMap** component to open the Map Editor.

2. If you selected **XML** in the **Accept Type** list of the **tRESTClient** component, define the input XML tree structure according to the service structure.
   a) In the input table, right-click the default *root* node of the *body* column, select **Rename** from the contextual menu, and rename it to *users*.

   b) Right-click the *users* node, select **Create Sub-Element** from the contextual menu, and create sub-element named *user*. Set *user* as the loop element because the XML structure of the Web service to be invoked is looped on this element.

   c) Right-click the *user* node, select **Create Attribute** from the contextual menu, and enter *id* in the **Create New Attribute** dialog box to create an attribute named *id* for the *user* node.

   d) Right-click the *user* node again, select **Create Sub-Element** from the contextual menu, and enter *first_name* in the **Create New Element** dialog box to create an sub-element named *first_name* for the *user* node.

   Repeat this operation to create another sub-element under the *user node*, *last_name*.

   e) Drop the *id*, *first_name* and *last_name* columns from the input table to the output table, and then click **OK** to validate the mapping and close the Map Editor.
If you selected JSON in the Accept Type list of the tRESTClient component, the response from the server end will be sent back in JSON format and converted to document type. In this example, the converted response structure looks like the following:

```xml
<root>
  <users>
    <user>
      <id>2</id>
      <first_name>Theodore</first_name>
      <last_name>Harding</last_name>
    </user>
  </users>
</root>
```

Note that the <root> element is removed if the Wrap JSON Response check box is cleared in the Advanced settings of the tRESTClient component.

Define the input XML tree structure accordingly and map it with the output data flow in a similar manner as described above.
3. Double-click the tLogRow component to open its **Basic settings** view.

4. Click the **Sync columns** button to make sure the component schema is synchronized with the output schema of the tXMLMap component.

5. In the **Mode** field, select the **Table** option to display the GET result in table cells.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Press **F6** or click **Run** on the Run console to launch the Job.

   The console shows that the tRESTClient component successfully reads the user information from the server end corresponding to the specified user ID.
If you selected XML in the Accept Type list of the tRESTClient component, the execution result will be:

```
[statistics] connecting to socket on port 3496
[statistics] connected
May 26, 2014 5:30:17 PM
org.apache.cxf.interceptor.LoggingOutInterceptor
INFO: Outbound Message
--------------------------
ID: 1
Address: http://localhost:3088/users?from=2&to=2
Http-Method: GET
Content-Type: */*
Headers: {Accept=[application/xml], Content-Type=[/*/*]}
--------------------------
May 26, 2014 5:38:17 PM
org.apache.cxf.interceptor.LoggingInInterceptor
INFO: Inbound Message
--------------------------
ID: 1
Response-Code: 200
Encoding: ISO-8859-1
Content-Type: application/xml
Headers: {content-type=[application/xml], Date=[Mon, 26 May 2014 09:38:17 GMT], Server=[Jetty(8.1.14.v20131031)], transfer-encoding=[chunked]}
Payload: <?xml version="1.0" encoding="UTF-8"?>
<users><user
id="2"><first_name>Theodore</first_name><last_name>Harding</last_name></user></users>
-------------------------------
<table>
<thead>
<tr>
<th>Show User Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>
-------------------------------
[statistics] disconnected
```

If you selected JSON in the Accept Type list of the tRESTClient component, the execution result will be:
Scenario 2: Updating user information by interacting with a RESTful service

This scenario describes a three-component Job that updates the information of a list of users to a remote database through a REST service using the HTTP POST method. When executed, the Job displays the client-server message exchange information on the Run console.

The user information to be updated to the server is stored in a CSV file, which looks like the following:

```
id;first_name;last_name
1;John;Smith
2;Martin;Reagan
3;James;White
```
Prerequisites:

If you are a *Talend Open Studio for ESB* user, create a Job as described in *Scenario 3: REST service accepting HTTP POST requests* on page 3227 and run the Job as a REST server to expose a REST service that accepts HTTP POST requests. Upon execution of the Job, the console displays the service implementation information, including the service endpoint URL, which is http://localhost:8045/users in this example. If you enter http://localhost:8045/users?_wadl in your Web browser, you should see the service definition information like the following:

```xml
<application>
  <grammars/>
  <resources base="http://localhost:8045/users">
    <resource path="/">
      <resource path="/post/(id)/{first_name}/" last_name" style="template" type="xs:string"/>
      <param name="id" style="template" type="xs:int"/>
      <param name="first_name" style="template" type="xs:string"/>
    </resource>
    <method name="POST">
      <response>
        <representation mediaType="application/xml"/>
        <representation mediaType="text/xml"/>
        <representation mediaType="application/json"/>
      </response>
    </method>
  </resources>
</application>
```

If you are not a *Talend Open Studio for ESB* user, then you need to get the service-related information from your REST service provider including the URL, the resource path, and the data structure, and make necessary modifications in the scenario configurations accordingly.

Setting up the Job

Procedure

1. Create a Job and add the following components by typing their names in the design workspace or drop them from the Palette:
• **tFileInputDelimited**, to read user information of a local file,
• **tXMLMap**, to adapt the input structure to the REST service structure, and
• **tRESTClient**, used to call the REST service to send data to the remote database.

2. Connect the **tFileInputDelimited** to the **tXMLMap** using a **Row > Main** connection.
3. Connect the **tXMLMap** to the **tRESTClient** using a **Row > Main** connection, and give the output flow a name, **request** in this example.
4. Label the components to best describe the actions to perform.

### Configuring the components

#### Configuring the input data and the structure mappings

**Procedure**

1. Double-click the **tFileInputDelimited** component to open its **Basic settings** view.

   ![Input(tFileInputDelimited_1)](image)

2. Specify the input file in the **File name** field, fill the **Header** field with 1 to skip the header row, and keep the rest parameters as they are.
3. Click the [...] button next to **Edit schema** to open the [Schema] dialog box, and edit the input schema as follows:
   - **id**, type Integer, 2 characters long, set as the key column
   - **first_name**, type String
   - **last_name**, type String
4. Double-click the **tXMLMap** component to open the Map Editor.

5. Rename the root node in the output table: right-click the node, select **Rename** from the contextual menu, and specify a new name in the pop-up dialog box, **user** in this example.

6. Select all the three columns in the input table and drop them onto the **user** node, and select the **Create as sub-element of target node** option from the pop-up dialog box to set these columns as sub-elements of the **user** node. When done, click **OK** to validate the mappings and close the Map Editor.
Configuring the service call

Procedure

1. Double-click the tRESTClient component to open its Basic settings view.

2. Fill the URL field with the URI location where the REST service is accessible, "http://localhost:8088/users" in this example.

3. In the Relative Path field, enter the resource path, "/post/" + row1.id + "/" + row1.first_name + "/" + row1.last_name in this example. This will send the data from the input row to the server end via the resource path.

4. From the HTTP Method list, select GET to send an HTTP request for retrieving the existing records. From the Accept Type list, select the type the client end is prepared to accept for the response from the server end, XML.

5. In the Advanced settings view of the tRESTClient component, select the Log messages check box to log the message exchange information with the server. Leave the rest of the settings as they are.

Running the Job and checking the result

Press Ctrl+S to save your Job, and press F6 to execute it.

The console displays the client-server exchange information:
The console of the server Job displays the exchange information and the database update result.
ID: 5
Encoding: ISO-8859-1
Http-Method: POST
Content-Type: application/xml
Headers: {Accept=[*/*], Cache-Control=[no-cache], connection=[keep-alive], Content-Length=[123], content-type=[application/xml], Host=[localhost:8045], Pragma=[no-cache], User-Agent=[Apache CXF 2.7.11]}
Payload: <?xml version="1.0" encoding="UTF-8"?>
  <user><id>5</id><first_name>Robert</first_name><last_name>Thomson</last_name></user>

-------------------------------
| tLogRow_1                   |
|-------------------------------
| id|first_name|last_name|
|-------------------------------
| 1 | John     | Smith    |
| 2 | Martin   | Reagan   |
| 3 | James    | White    |
| 4 | Jenny    | Jackson  |
| 5 | Robert   | Thomson  |

[INFO ]: org.apache.cxf.interceptor.LoggingOutInterceptor - Outbound Message
-------------------------------
ID: 5
Response-Code: 200
Content-Type: application/xml
Headers: {Content-Type=[application/xml], Date=[Thu, 29 May 2014 09:13:02 GMT]}
Payload: <?xml version="1.0" encoding="UTF-8"?>
  <user><id>5</id><first_name>Robert</first_name><last_name>Thomson</last_name></user>
tRESTRequest

Receives GET/POST/PUT/PATCH/DELETE requests from the clients on the server end.

tRESTRequest accepts the HTTP and/or HTTPS requests from the clients and supports GET, POST, PUT, PATCH and DELETE HTTP methods.

Note:
To enable the HTTPS support, you have to generate a keystore and add some HTTPS security configuration properties in the org.ops4j.pax.web.cfg file of your Runtime container before deploying the service on it. For more information, see the Talend ESB Container Administration Guide.

tRESTRequest Standard properties

These properties are used to configure tRESTRequest running in the Standard Job framework.

The Standard tRESTRequest component belongs to the ESB family.

This component is relevant only when used with one of the Talend solutions with ESB, as it should be used with the Service Repository node and the Data Service creation related wizard(s).

Basic settings

<table>
<thead>
<tr>
<th>REST Endpoint</th>
<th>Fill this field with the URI location where REST-ful web service will be accessible for requests.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Note:</td>
</tr>
<tr>
<td></td>
<td>If you want your service to be available on both HTTP and HTTPS, fill the field with a relative path.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REST API Mapping</th>
<th>Click the [+] button beneath the mapping table to add lines to specify HTTP request:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Output Flow: Click the [...] button to specify the name of an output flow and set the schema for that output flow in the dialog box afterwards.</td>
</tr>
<tr>
<td></td>
<td>The schema is not mandatory, so if you do not need to pass additional parameters to the tRESTRequest component, you can leave the schema empty. However, you will have to populate the schema if you have URI Path parameters set in the URI Pattern field or if you need to add optional request parameters such as URI Query, HTTP Header or Form parameters, to the URI specified in the REST Endpoint field.</td>
</tr>
<tr>
<td></td>
<td>Add a schema with the name body to get the request body of POST and PUT methods. It supports Document, String, and Byte[] types.</td>
</tr>
<tr>
<td></td>
<td>If you specify URI parameters in the output flow schema, you might need to define what type of parameter it is in the Comment field of the schema. By default, if you leave the Comment field empty, the</td>
</tr>
</tbody>
</table>
The parameter is considered as a Path parameter. Below is a list of supported Comment values:

- empty or path corresponds to the default @PathParam,
- query corresponds to @QueryParam,
- form corresponds to @FormParam,
- header corresponds to @HeaderParam.
- matrix corresponds to @MatrixParam.
- multipart corresponds to the CXF specific @Multipart, representing the request body. It can be used only with POST and PUT HTTP methods.

Note:
We recommend you to set the default values of your optional parameters (Header, Query, Form). To do so, fill in the Default columns of the schema.

HTTP Verb: Select a HTTP method (GET/POST/PUT/PATCH/DELETE/OPTIONS/HEAD) from the list.

URI pattern: Fill this field with REST-ful URIs that describe the resource.

Consumes: Select the format type of the consume content that the component will use from XML or JSON, XML, JSON, Form, Multipart and Any in the list when the HTTP method is POST, PUT or PATCH.

Produces: When the HTTP method is GET, POST, PUT, PATCH, or DELETE, select the format type of the produce content that the component will use from XML or JSON, XML, JSON, HTML and Any, or select <oneway> in the list to accept one way requests.

Streaming: Select this check box to stream the response data by chunks so that the large volumes of data can be processed efficiently.

Use Service Locator: Select this check box to enable the Service Locator. It maintains the availability of the service to help meet demands and service level agreements (SLAs). Specify the Service namespace and the Service name in the corresponding fields.

Use Service Activity Monitor: Select this check box to enable the Service Activity Monitor. It captures events and stores this information to facilitate in-depth analysis of service activity and track-and-trace of messages throughout a business transaction. This can be used to analyze service response times, identify traffic patterns, perform root cause analysis and more.

Use Authentication (ESB runtime only): Select this check box to enable the authentication option for the current service. This option works in runtime only. Select the authentication type from:
- **Basic HTTP**: The simplest technique for enforcing access controls to web resources using standard fields in the HTTP header.
- **SAML Token**: An XML-based, open-standard data format for exchanging authentication and authorization data between an identity provider and a service provider.

### Use Business Correlation

Select this check box to enable the correlation option so that chained service calls will be grouped under the same correlation ID. tRESTRequest will extract the correlation ID from the request header and store it in the component variable for further use in the flow.

If this option is not enabled on the client side, a correlation ID will be generated automatically in tRESTRequest.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Log messages</strong></td>
<td>Select this check box to log the message exchange between the service provider and the consumer.</td>
</tr>
<tr>
<td><strong>Wrap JSON Request</strong></td>
<td>Select this check box to wrap the JSON request with a root element.</td>
</tr>
<tr>
<td><strong>Convert JSON values to String in response</strong></td>
<td>Select this check box to convert the JSON values to string format in the response.</td>
</tr>
<tr>
<td><strong>Service Locator Customer Properties</strong></td>
<td>This option appears when Use Service Locator is enabled in the Basic settings tab. Click [+] to add as many properties as needed to the table. Enter the name and the value of each property in the Property Name field and the Property Value field respectively to identify the service.</td>
</tr>
<tr>
<td><strong>Service Activity Customer Properties</strong></td>
<td>This option appears when Use Service Activity Monitor is enabled in the Basic settings tab. Click [+] to add as many properties as needed to the table. Enter the name and the value of each property in the Property Name field and the Property Value field respectively to identify the service.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong></td>
<td>the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>URI</strong></td>
<td>the URI of the REST request. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td><strong>URI_BASE</strong></td>
<td>the base URI of the REST request. This is a Flow variable and it returns a string.</td>
</tr>
</tbody>
</table>
### Usage

#### Usage rule

This component covers the possibility that a **Talend** Job can be wrapped as a service, with the ability to input a request to a service into a Job and return the Job result as a service response.

The **tRESTRequest** component should be used with the **tRESTResponse** component to provide a Job result as a response, in case of a request-response communication style.

#### Limitation

Using context variables for dynamic endpoint or Service Locator namespace works in the Studio only. It is not supported in Runtime.

---

### Scenario 1: REST service accepting HTTP GET requests and sending responses

This scenario applies only to a Talend solution with ESB.

This scenario describes the process of accepting an HTTP request from the client, processing it and sending the response back.
Configuring the tRESTRequest component

Procedure

1. Drop the following components from the Palette onto the design workspace: a tRESTRequest, two tXMLMap and two tRESTResponse.

2. Double-click tRESTRequest in the design workspace to display its Basic settings view.

3. Fill the REST Endpoint field with the URI location where the REST-ful web service will be accessible for requests. For example, "http://192.168.0.235:8088/user".

   **Note:**
   If you want your service to be available on both HTTP and HTTPS, fill the field with a relative path. For example, if you type in "/test", your service will be available on both http://<DefaultHTTPEnpointAddress>/test and https://<DefaultHTTPSEnpointAddress>/test, provided that you have configured your Runtime container to support HTTPS. For more information, see the Talend ESB Container Administration Guide.

4. Click the [+] button to add one line in the REST API Mapping table.
5. Select the newly-added line and click the [...] button in the **Output Flow** column to add a schema for the output flow.

In this scenario, the output flow will be named as *GetOneUser*.

<table>
<thead>
<tr>
<th>Column</th>
<th>Key</th>
<th>Type</th>
<th>N.</th>
<th>Date Patter</th>
<th>Length</th>
<th>Prec.</th>
<th>De.</th>
<th>Co.</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td></td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Then click the [+] button to add a new line *id* to the schema in the dialog box.

6. Click **OK** to save the schema.

7. Select **GET** from the list in the **HTTP Verb** column.

8. Fill the field in the **URI Pattern** column with "/{id}/".

9. In the same way, add another line in the **REST API Mapping** table and name this schema *GetUserNumber*. Add a line *string* of *String* type to the schema. Enter *query* in the **Comment** field to get the number parameter in the request.

Select **GET** from the list in the **HTTP Verb** column. Fill the field in the **URI Pattern** column with "/number".

**Configuring the first tXMLMap component**

**Procedure**

1. Connect **tRESTRequest** to a **tXMLMap** using the **Row > GetOneUser** connection.

2. Double-click **tXMLMap** in the design workspace to open the **Map Editor**.
3. Click the [+] button on the top right to add an output and name it as `ResponseUsers`.
4. Click the [+] button on the bottom right to add two columns for the output.
   - Name the first column as `body` and set the Type to Document.
   - Name the second column as `string` and set the Type to String.
5. Right-click on the node `root` and select Create Sub-Element to create a sub-element. Name the sub-element as `foo` in the popup dialog box.
6. Right-click on the `foo` node created in the previous step and select As loop element.
7. Select the `id` column of the `GetOneUser` table to the left and drop it onto the Expression field of the `foo` node of the `ResponseUsers` table to the right.
8. Click OK to save the settings.

**Configuring the second tXMLMap component**

**Procedure**
1. Connect `tRESTRequest` to the other `tXMLMap` using the Row > GetUserNumber connection.
2. Double-click the first `tXMLMap` in the design workspace to open the Map Editor.
3. Click the [+] button on the top right to add an output and name it as `ResponseUserNumber`.

4. Click the [+] button on the bottom right to add one column for the output. Name the first column as `body` and set the Type to Document.

5. Right-click on the node `root` and select Create Sub-Element to create a sub-element. Name the sub-element as `number` in the popup dialog box.

6. Right-click on the `number` node created in the previous step and select As loop element.

7. Select the `string` column of the `GetUserNumber` table to the left and drop it onto the Expression field of the `number` node of the `ResponseUserNumber` table to the right.

8. Click OK to save the settings.

**Configuring the tRESTResponse component**

**Procedure**

1. Connect `tXMLMap` to a `tRESTResponse` using Row > ResponseUsers connection. The schema defined in `tXMLMap` is retrieved in `tRESTResponse` automatically.
2. Select **OK(200)** from the **Return status code** list.
3. Leave the rest of the settings as they are.
4. Connect **tRESTRequest** to the other **tRESTResponse** using the **Row > GetUserNumber** connection.

### Saving and executing the Job

**Procedure**

1. Save the Job and press **F6** to execute it.

   ```
   [statistics] connecting to socket on port 3888
   [statistics] connected
   [INFO ]: org.apache.cxf.endpoint.ServerImpl - Setting the server's publish address to be http://localhost:8088/user
   [INFO ]: org.eclipse.jetty.server.AbstractConnector - Started
   SelectChannelConnector@localhost:8088
   ```

2. Go to your browser if you want to test the service.

   ![Firefox](http://localhost:8088/users/1)
   ![Firefox](http://localhost:8088/users/fuo)

   The HTTP request for a user id is accepted by the REST service and the HTTP response is sent back to the server.

3. Repeat this step and type in **http://localhost:8088/user/number?string=123** in the address bar. Press **Enter**.

   You can see that **123** is returned in the response.
Scenario 2: Using URI Query parameters to explore the data of a database

This scenario applies only to a Talend solution with ESB.

This scenario describes how to use URI query parameters in tRESTRequest to explore data of a database, and send the response via the tRESTResponse.

To do so, you can create two subjobs linked together by an OnSubjobOk connection; this way the two subjobs will be executed sequentially. For more information on Trigger connection, see the Talend Studio User Guide. The first subjob will create and populate the database and the second one will allow to explore the database through the REST service.

Creating the first subjob

About this task

To do this, proceed as follows:

Procedure

1. Drop the following components from the Palette onto the design workspace: tFixedFlowInput from the Misc family and tMysqlOutput from the Databases > Mysql family.
2. Link tFixedFlowInput to tMysqlOutput using a Row > Main connection.
3. Double-click **tFixedFlowInput** to display its **Basic settings** view:

![tFixedFlowInput settings](image)

4. Click the [...] button next to **Edit schema** to open the schema editor.

![Schema editor](image)

5. In the schema editor, click the [+] button three times to add three lines and set them as displayed in the above screenshot.

6. Click **Ok**.

7. Back to **tFixedFlowInput Basic settings** view, in the **Mode** area, select the **Use inline table** option.

8. Under the inline table, click the [+] button three times to add three rows in the table.

9. In the inline table, click the **id** field of the first row and type in 1.

10. Click the **firstname** field of the first row, press Ctrl+Space to display the autocompletion list and select the **TalendDataGenerator.getFirstName()** variable in the list.

11. Click the **lastname** field of the first row, press Ctrl+Space to display the autocompletion list and select the **TalendDataGenerator.getLastName()** variable in the list.

12. Do the same for the two following rows to obtain the settings displayed in the screenshot.
13. Double-click **tMysqlOutput** to display its **Basic settings** view:

![tMysqlOutput](image)

14. From the **Property Type** list, leave **Built-in** and fill in the **Host**, **Port**, **Database**, **Username** and **Password** fields manually. If you centralized your connection information to the database in the **Metadata > DB Connections** node of the **Repository**, you can select **Repository** from the list and the fields will be automatically filled in.

For more information about storing metadata, see *Talend Studio User Guide*.

15. In the **Table** field, type in the name of the table in which the data will be loaded, for example: **users**.

16. From the **Action on table** list, select **Drop table if exists and create**, select **Insert** from the **Action on data** list.

17. Click **Sync columns** to retrieve the schema coming from the previous component.

**Creating the second subjob**

The following components are used as displayed in the first screenshot:

- **tRESTRequest** and **tRESTResponse** from the **ESB > REST** family,
- **tFlowToIterate** from the **Orchestration** family,
- **tMysqlInput** from the **Databases > Mysql** family,
- **tXMLMap** from the **Processing** family.

**Configuring the tRESTRequest component**

**About this task**

To do this, proceed as follows:
Procedure

1. Double-click `tRESTRequest` in the design workspace to display its **Basic settings** view:

![tRESTRequest Basic settings](image)

2. Fill the **REST Endpoint** field with the URI location where the REST-ful web service will be accessible for requests. For example, "http://localhost:8088/users".

3. Click the `[+]` button to add one line in the **REST API Mapping** table.

4. Select the newly-added line and click the `[...]` button in the **Output Flow** column to add a schema for the output flow.

5. In the dialog box, name the output flow `getUsers`. A schema editor dialog box appears.

![Schema editor dialog box](image)

6. In the schema editor, click the `[+]` button twice to add two lines and set them as displayed in the above screenshot.

7. Click **OK**.

8. Back to `tRESTRequest Basic settings` view, select **GET** from the list in the **HTTP Verb** column.

9. Leave the **URI Pattern** column as is.

10. Use the corresponding link to connect to the following component: connect `tRESTRequest` to `tFlowToIterate` using **Row > getUsers** connection.
11. Leave the tFlowToIterate settings as is.
12. Connect tFlowToIterate to tMysqlInput using Row > Iterate connection.

**Configuring the tMysqlInput component**

**About this task**

To do this, proceed as follows:

**Procedure**

1. Double-click tMysqlInput to display its Basic settings view:

   ![tMysqlInput_1](image)

   1. From the **Property Type** list, leave **Built-in** and fill in the **Host**, **Port**, **Database**, **Username** and **Password** fields manually. If you centralized your connection information to the database in the Metadata > DB Connections node of the Repository, you can select **Repository** from the list and the fields will be automatically filled in.

      For more information about storing metadata, see Talend Studio User Guide.

   2. Leave the **Schema** list as **Built-in** and click the [...] button next to the **Edit schema** field.

   3. In the schema editor, define the schema exactly like the one of the tFixedFlowInput.

   4. Leave the **Query Type** list as **Built-in** and fill in the **Query** field with the following SQL query allowing to explore the database data with the URI query set in the tRESTRequest component:

      "select * from users where id >= " + globalMap.get("getUsers.from") + " and id <= " + globalMap.get("getUsers.to")

**Configuring the tXMLMap component**

**Procedure**

1. Right-click tMysqlInput, hold and drag to tXMLMap to connect the two components together.
2. Double-click **tXMLMap** in the design workspace to open the **Map Editor**.

3. Click the [+] button on the top right to add an output and name it as **ResponseUsers**.
4. Click the [+] button on the bottom right to add two columns for the output.
   - Name the first column as **body** and set the **Type** to **Document**.
   - Name the second column as **string** and set the **Type** to **String**.
5. Right-click on the **root** node, select **Rename** in the list and rename it **users**.
6. Right-click on the **users** node and select **Create Sub-Element** to create a sub-element. Name the sub-element **user** in the popup dialog box.
7. Right-click on the **user** node created in the previous step and select **As loop element**.
8. Select the **id** column of the **row2** table to the left and drop it onto the **user** node of the **ResponseUsers** table to the right.
9. In the [Selection] dialog box, select the **Create as attribute of target** node option and click **OK**.
10. Select the **firstname** and **lastname** columns of the **row2** table to the left and drop it onto the **user** node of the **ResponseUsers** table to the right.
11. In the [Selection] dialog box, select the **Create as sub-element of target node** option and click **OK**.

12. Click the wrench icon on the top of the **ResponseUsers** table to open the setting panel.

13. Set the **All in one** feature as **true**, this way all XML data is outputted in one single flow.

14. Click **OK** to save the settings.

**Configuring the tRESTResponse component**

**Procedure**

1. Connect **tXMLMap** to **tRESTResponse** using **Row > ResponseUsers** connection.

2. The schema defined in **tXMLMap** is retrieved in **tRESTResponse** automatically. Leave the other settings as they are.
Connecting the two subjobs

About this task

Now that the two subjobs are created, you can connect them together:

Procedure

1. Right-click the \texttt{tFixedFlowInput} component of the first subjob.
2. Select \texttt{Trigger > OnSubjobOk} on the list.
3. Click the \texttt{tRESTRequest} component of the second subjob.

Results

This way, when executing the job, the second subjob will be executed only if the first one’s execution succeeded.

Saving and executing the Job

Procedure

1. Save the Job and press \texttt{F6} to execute it.

```
[statistics] connecting to socket on port 4020
[statistics] connected
[INFO ]: org.apache.cxf.endpoint.ServerImpl - Setting the server's publish address to be http://localhost:8088/users
[INFO ]: org.eclipse.jetty.server.AbstractConnector - Started
SelectChannelConnector:localhost:8088
```

2. Go to your browser if you want to test the service.
   For example, use the URI query \texttt{?to=2} to retrieve the data of the two first users.
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```xml
- <users>
  - <user id="1">
    <firstname>George</firstname>
    <lastname>Ford</lastname>
  </user>
  - <user id="2">
    <firstname>Rutherford</firstname>
    <lastname>Coolidge</lastname>
  </user>
- </users>
```

The HTTP request for a user id is accepted by the REST service and the HTTP response is sent back to the server.

For a use case that calls this REST service using Talend Open Studio for ESB components, see Scenario 1: Getting user information by interacting with a RESTful service on page 3197.

**Scenario 3: REST service accepting HTTP POST requests**

This scenario applies only to a Talend solution with ESB.

This scenario describes a Job composed of two subjobs: the first subjob exposes a REST service that accepts HTTP POST requests from REST clients, writes data to a database upon receiving an HTTP request, and displays the server-client exchange information on the Run console; the second subjob displays the results of database updates.
**Setting up the Job**

**Procedure**

1. Create a Job and add the following components to the Job by typing their names in the design workspace or dropping them from the Palette: a tRESTRequest, a tMysqlOutput, a tXMLMap, and a tRESTResponse.

2. Connect the tRESTRequest to the tMysqlOutput component using a Row > Main connection, and give the output flow a name, request in this example.

3. Connect the tMysqlOutput component to the tXMLMap component using a Row > Main connection.

4. Connect the tXMLMap component to the tRESTResponse component using a Row > Main connection, and give it a name, response in this example. Click OK in the pop-up dialog box to accept the schema propagation from the tRESTResponse component.

5. Add a tMysqlInput component and a tLogRow component to the Job by typing their names in the design workspace or dropping them from the Palette, and connect the tMysqlInput to the tLogRow using a Row > Main connection.

6. Connect the tMysqlOutput to the tMysqlInput using a Trigger > OnComponentOk connection to link the two subjobs.

**Configuring the components**

**Configuring REST request parameters**

**Procedure**

1. Double-click the tRESTRequest component to open its Basic settings view.

   ![tRESTRequest settings](image)

2. Fill the REST Endpoint field with the URI location where the REST service will be accessible for requests, "http://localhost:8045/users" in this example.

3. Click the output flow name, which is request in this example, in the Output Flow column of the REST API Mapping table to show the [...] button, and click the button to open the [Schema] dialog box. Then define the schema for the request flow as follows:
   - id, type Integer, 2 characters long, set as the key column
   - first_name, type String
   - last_name, type String
When done, click OK to validate the schema setting and close the dialog box.

4. Click the HTTP Verb column and select POST from the list.

5. In the URI Pattern column, enter the acceptable URI pattern of POST requests, "/post/{id}/{first_name}/{last_name}" in this example. Leave the rest parameters as they are.

6. Click the Advanced settings tab, and select the Log messages check box.

Configuring the database and the response parameters

Procedure

1. Double-click the tMysqlOutput component to open its Basic settings view.

2. Leave the Property Type as Built-in and fill in the database connection details manually in the Host, Port, Database, Username and Password fields.

If you have centralized your connection information to the database in the Metadata > DB Connections node of the Repository, you can select Repository from the Property Type list and browse to the centralized connection to have the fields automatically filled in. For more information about storing metadata, see Talend Studio User Guide.
3. Fill in the **Table** field with the target database table name, *users* in this example.

4. Select the actions to be carried out on the database table and data according to your needs from the corresponding lists. In this example, the target table will be created if it does not exist in the specified database, and data from client requests will be inserted, or updated if it already exists, to the database table.

5. Click the **Sync columns** button to synchronize the table schema with the input schema.

6. Double-click the **tXMLMap** component to open the Map Editor.

7. Rename the root node in the output table: right-click the node, select **Rename** from the contextual menu, and specify a new name in the pop-up dialog box, *user* in this example.

8. Select all the three columns in the input table and drop them onto the *user* node, and select the **Create as sub-element of target node** option from the pop-up dialog box to set these columns as sub-elements of the *user* node. When done, click **OK** to validate the mappings and close the Map Editor.

9. Double-click the **tRESTResponse** component and set the response parameters according to your needs. In this example, we keep the default settings for all the parameters.

**Configuring result display**

**Procedure**

1. Double-click the **tMysqlInput** component to open its **Basic settings** view.
2. Specify the database connection details, table name and table schema, which are the same as in the tMysqlOutput component.

3. Click the Guess Query button to fill the Query field with the query statement.

4. Double-click the tLogRow component and select the Table option to display the content of the database in table cells.

**Saving the executing the Job**

**Procedure**

1. Press Ctrl+S to save your Job.

2. Click the Run button on the Run tab or press F6 to run it.

The console displays the service implementation information, including the service URL.

When an HTTP POST request is received from a client, the console displays the relevant exchange information and the database update result.
Scenario 4: REST service accepting HTTP POST requests and sending responses

This scenario applies only to a Talend solution with ESB.

This scenario describes how to build a REST service that accepts HTTP POST requests of Document type from the client, processes it and sends the response back. To do this, two Jobs are built: one to create the REST service and the other to call it.

Creating the REST service Job

Drag and drop the following components from the Palette onto the design workspace: a tRESTRequest, a tXMLMap and a tRESTResponse.
Configuring the tRESTRequest component

Procedure

1. Double-click **tRESTRequest** in the design workspace to display its **Basic settings** view.

2. In the **REST Endpoint** field, keep the default URI "http://localhost:8088/" where the REST-ful web service will be accessible for requests.

3. Click the [+] button to add one line in the **REST API Mapping** table.

4. Select the newly-added line and click the [...] button in the **Output Flow** column to add a schema for the output flow. Give the name for the schema **UserInfo** in this scenario.
   
The schema editor opens. Click the [+] button to add a new line **body of Document type** and click **OK** to save the schema.
5. Back to tRESTRequest Basic settings view, select POST from the list in the HTTP Verb column.
6. Fill the field in the URI Pattern column with "/UserInfo". Keep the default settings of the other options.

Configuring the body of the response

Procedure

1. Connect tRESTRequest to tXMLMap using the Row > UserInfo connection.
2. Double-click tXMLMap in the design workspace to open the Map Editor.
3. Right-click on the root node of the input table and select Create Sub-Element. Name the sub-element as person in the popup dialog box.

4. Right-click on the person node created in the previous step and select As loop element.

5. Create two sub-elements to the person node by selecting Create Sub-Element in the contextual menu, id and name in this use case.

6. Click the [+] button on the top right to add an output and name it as Response.

7. Click the [+] button on the bottom right to add a column body of Document type for the output.

8. Right-click the node root in the output table and select Create Sub-Element. Name the sub-element as person in the popup dialog box.

9. Right-click the person node created in the previous step and select As loop element.

10. Create three sub-elements to the person node by selecting Create Sub-Element in the contextual menu, id, name and company in this use case.

11. Select the id column from the input table and drop it onto the Expression field of the id node of the output table.

12. Select the name column from the input table and drop it onto the Expression field of the name node in the output table.

   Click the Expression field of the name node in the output table, and click [...] in this field to show the [Expression Builder] wizard.

   In the Expression area, enter StringHandling.UPCASE([UserInfo.body:/root/person/name]) to convert the name of the user to upper case. Click OK to close the wizard.

   ![Expression Builder](image)

   For further information about using the Expression Builder, see Talend Studio User Guide.

13. In the Expression field of the company node, enter "Talend".

14. Click OK to save the settings.

Configuring the tRESTResponse component

Procedure

1. Connect tXMLMap to a tRESTResponse using Row > Response connection.

   The schema defined in tXMLMap is retrieved in tRESTResponse automatically.
2. Select **OK(200)** from the Return status code list.

3. Leave the rest of the settings as they are.

**Launching the service**

Save the Job and press **F6** to execute it. The service is started.

**Creating the consumer Job**

Drag and drop the following components from the Palette onto the design workspace: a \texttt{tFixedFlowInput}, a \texttt{tXMLMap}, a \texttt{tRESTClient} and a \texttt{tLogRow}.

**Configuring the \texttt{tFixedFlowInput} component**

**Procedure**

1. Double-click \texttt{tFixedFlowInput} in the design workspace to display its Basic settings view.
2. Click **Edit schema** to show the schema editor. In the schema editor, click the [+ ] button to add two columns of **String** type, **id** and **name**. Click **Ok** to close the schema editor.

3. Back to **tFixedFlowInput Basic settings** view, in the **Mode** area, select the **Use Inline Content** option.

4. In the **Content** box, enter the **id** and **name** of three users, for example:

```
1;ford
2;smith
3;thomson
```

**Configuring the tXMLMap component**

**Procedure**

1. Connect **tFixedFlowInput** to **tXMLMap** using the **Row > Main** connection.

2. Connect **tXMLMap** to **tRESTClient** using the **Row > Main** connection, and name the output flow **request** in this example.

3. Double-click **tXMLMap** in the design workspace to open the **Map Editor**.
4. Right-click on the node root in the output table and select Create Sub-Element. Name the sub-element as person in the popup dialog box.

5. Right-click on the person node created in the previous step and select As loop element.

6. Create two sub-elements to the person node by selecting Create Sub-Element in the contextual menu, id, and name in this use case.

7. Select the id column from the input table and drop it onto the Expression field of the id node of the output table.

8. Select the name column from the input table and drop it onto the Expression field of the name node in the output table.

9. Click OK to save the settings.

Configuring the tRESTClient component

Procedure

1. Double-click the tRESTClient component to open its Basic settings view.
2. Fill the **URL** field with the URL of the REST service built in the service Job, “http://localhost:8088/”.
3. Fill the **Relative Path** with the “UserInfo”.
4. From the **HTTP Method** list, select **POST**.
5. Leave the rest of the settings as they are.
6. Connect **tRESTClient** to **tLogRow** using the **Row > Response** connection. Keep the default settings of the **tLogRow** to monitor the message exchanges.

**Saving and executing the Job**

Save the Job and press **F6** to execute it. The user information is shown in the console, including **id**, **name**, and **company**. The name of the user has been converted to upper case.
Scenario 5: REST service accepting HTTP POST requests in an HTML form

This scenario applies only to a Talend solution with ESB.

This scenario describes how to upload data to a REST service using HTTP POST in a HTML form file. To do so, a Job is created to accept HTTP POST requests using the tRESTRequest component.

A predefined HTML form UploadFile.html is used to upload data to the REST service:

```
<form action="http://localhost:8088/UserInfo" method="post" enctype="multipart/form-data">
  <input type="file" name="id" />
  <button>upload</button>
</form>
```

The customer.xml that contains a user ID will be uploaded:

```
<customer>
  <id>100</id>
</customer>
```

Setting up the Job

Drop the following components from the Palette onto the design workspace: a tRESTRequest, a tJavaRow, a tXMLMap and a tRESTResponse.
Configuring the tRESTRequest component

Procedure

1. Double-click tRESTRequest in the design workspace to display its Basic settings view.

2. In the REST Endpoint field, keep the default URI "http://localhost:8088/" where the REST-ful web service will be accessible for requests.

3. Click the [+ ] button to add one line in the REST API Mapping table.

4. Select the newly-added line and click the [...] button in the Output Flow column to add a schema for the output flow. Give the name for the schema UserInfo in this scenario.

   The schema editor opens. Click the [+ ] button to add a new line id of byte[] type and enter multipart in the Comment field. Click OK to save the schema.

5. Back to tRESTRequest Basic settings view, select POST from the list in the HTTP Verb column.

6. Fill the field in the URI Pattern column with "/UserInfo".

7. Select Multipart in the Consumes list. Keep the default settings of the other options.
Configuring the tJavaRow component

Procedure

1. Connect tRESTRequest to tJavaRow using the Row > UserInfo connection.
2. Double-click tJavaRow in the design workspace to display its Basic settings view.

3. Enter the following code in the Code box to convert the message into Document type and print it in the console.

```java
String result = new String(input_row.id, 0, input_row.id.length);
routines.system.Document doc = new routines.system.Document();
doc.setDocument(org.dom4j.DocumentHelper.parseText(result));
output_row.id = doc;
System.out.println(result);
```

Configuring the tXMLMap component

Procedure

1. Connect tJavaRow to tXMLMap using the Row > Main connection.
2. Double-click tXMLMap in the design workspace to open the Map Editor.
3. Click the [+] button on the bottom left to add a column id of Document type for the input.

4. Right-click the root node of the input table and select Rename in the contextual menu and rename it customer.

5. Create a sub-element to the customer node by selecting Create Sub-Element in the contextual menu. Name the sub-element as id in the pop-up dialog box.

6. Click the [+] button on the top right to add an output and name it as response.

7. Click the [+] button on the bottom right to add a column body of Document type for the output.

8. Right-click the root node of the input table and select Rename in the contextual menu and rename it customer.

9. Create a sub-element to the customer node by selecting Create Sub-Element in the contextual menu. Name the sub-element as id in the pop-up dialog box.

10. Select the id column from the input table and drop it onto the Expression field of the id node of the output table.

11. Click OK to save the settings.

**Configuring the tRESTResponse component**

**Procedure**

1. Connect tXMLMap to a tRESTResponse using Row > response connection.

   The schema defined in tXMLMap is retrieved in tRESTResponse automatically.
2. Select **OK(200)** from the **Return status code** list.
3. Leave the rest of the settings as they are.

**Saving and executing the Job**

**Procedure**

1. Save the Job and press **F6** to execute it.

2. Run the **UploadFile.html** using a web browser.

3. Click the **Browse** button to navigate to the **customer.xml** that contains the user ID and click **upload**.
4. The HTTP POST request is accepted by the REST service and the user ID is sent back to the server.

You can also view the response in the page source.

5. The content of the input file is printed in the execution console.
[statistics] connecting to socket on port 3005
[statistics] connected
Aug 18, 2015 5:35:18 PM org.apache.cxf.endpoint.ServerImpl initDestination
INFO: Setting the server's publish address to be
http://localhost:8080/
2015-08-18 17:35:48.376 INFO o.e.s.AbstractConnector:Started
SelectChannelConnector@localhost:8080
<?xml version="1.0" encoding="ISO-8859-15"?>

<customer>
  <id>100</id>
</customer>
tRESTResponse

Returns a specific HTTP status code to the client end as a response to the HTTP and/or HTTP requests.

**Note:**
To enable the HTTPS support, you have to generate a keystore and add some HTTPS security configuration properties in the `org.ops4j.pax.web.cfg` file of your Runtime container. For more information, see the *Talend ESB Container Administration Guide*.

### tRESTResponse Standard properties

These properties are used to configure tRESTResponse running in the Standard Job framework. The Standard tRESTResponse component belongs to the ESB family.

*This component is relevant only when used with one of the Talend solutions with ESB, as it should be used with the Service Repository node and the Data Service creation related wizard(s).*

#### Basic settings

<table>
<thead>
<tr>
<th>Return Body Type</th>
<th>Select a response body type from Document, String, and Byte[] in the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return status code</td>
<td>Select a status code from the list to indicate the request status.</td>
</tr>
<tr>
<td></td>
<td>- &lt;&lt;Custom&gt;&gt;: This option allows you to customize the status code. Enter the status code of your choice in the field.</td>
</tr>
<tr>
<td></td>
<td>- Bad Request (400): The request had bad syntax or was inherently impossible to be satisfied.</td>
</tr>
<tr>
<td></td>
<td>- Internal Server Error (500): The server encountered an unexpected condition which prevented it from fulfilling the request.</td>
</tr>
<tr>
<td></td>
<td>- OK (200): The request was fulfilled.</td>
</tr>
<tr>
<td></td>
<td>- Resource Not Found (404): The server has not found anything matching the URI given.</td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description, that is, it defines the number of fields to be processed and passed on to the next component.</td>
</tr>
<tr>
<td></td>
<td>The schema mode is Built-in: it is created and stored locally for this component.</td>
</tr>
<tr>
<td></td>
<td>Click Edit schema to make changes to the schema.</td>
</tr>
<tr>
<td></td>
<td>The input flow is retrieved from the previous component connected to tRESTResponse. Any change of the input flow in tRESTResponse will also be propagated to the previous component automatically.</td>
</tr>
</tbody>
</table>
The output flow of **tRESTResponse** is a hard-coded field named **body**, the type of which is defined in the **Return Body Type** list.

### Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Response Headers</strong></td>
<td>Type in the name-value pair(s) to pass additional information about the response. For more information about response headers, see the site <a href="http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6.2">http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6.2</a>.</td>
</tr>
<tr>
<td><strong>Drop JSON Response Root</strong></td>
<td>Select this check box to drop the root JSON element in the response.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Global Variables</strong></td>
<td><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <a href="http://www.talend.com">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage rule</strong></td>
<td>This component covers the possibility that a Talend Job can be wrapped as a service, with the ability to input a request to a service into a Job</td>
</tr>
</tbody>
</table>
**tRiakBucketList**

Retrieves a list of buckets from a Riak cluster and iterates on it.

**tRiakBucketList Standard properties**

These properties are used to configure tRiakBucketList running in the Standard Job framework.

The Standard tRiakBucketList component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. If you select the Use an existing connection check box, the Nodes table will not be available.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>Fill in the information required for a connection to a Riak cluster. • <strong>Host</strong>: enter the IP address or hostname of a Riak node in the Riak cluster that you want to connect to. • <strong>Port</strong>: enter the listening port number of a Riak node. You can leave this field empty to use its default value 8098. • <strong>Riak path</strong>: enter the Riak path for accessing a Riak node. You can leave this field empty to use its default value <code>riak</code>. One example of the URL for accessing a Riak node is <code>http://127.0.0.1:8098/riak</code>. For more information about the concepts related to Riak, see <a href="http://docs.basho.com/riak/latest/theory/concepts/">http://docs.basho.com/riak/latest/theory/concepts/</a>.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | **CURRENT_BUCKET_NAME**: the current bucket name. This is a Flow variable and it returns a string. **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. |
**Usage**

| Usage rule                  | This component can be used as a standalone component. |

**Related scenarios**

No scenario is available for the Standard version of this component yet.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*. 
tRiakClose

Closes an active connection to a Riak cluster so as to release occupied resources.

**tRiakClose Standard properties**

These properties are used to configure tRiakClose running in the Standard Job framework.

The Standard tRiakClose component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

| Component List | Select an active connection to a Riak cluster to be closed. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |

**Usage**

| Usage rule | This component is generally used with other Riak components, particularly tRiakConnection. |

**Related Scenario**

For a scenario in which tRiakClose is used, see *Scenario: Exporting data from a Riak bucket to a local file* on page 3255.
tRiakConnection

Opens and reuses of the connection it creates to a Riak cluster.

**tRiakConnection Standard properties**

These properties are used to configure tRiakConnection running in the Standard Job framework.

The Standard tRiakConnection component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| Nodes | Fill in the information required for a connection to a Riak cluster.

- **Host**: enter the IP address or hostname of a Riak node in the Riak cluster that you want to connect to.
- **Port**: enter the listening port number of a Riak node.
  You can leave this field empty to use its default value 8098.
- **Riak path**: enter the Riak path for accessing a Riak node.
  You can leave this field empty to use its default value riak.

One example of the URL for accessing a Riak node is http://127.0.0.1:8098/riak.

For more information about the concepts related to Riak, see http://docs.basho.com/riak/latest/theory/concepts/.

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | This component is generally used with other Riak components, particularly *tRiakClose*. |

### Related scenario

For a scenario in which *tRiakConnection* is used, see *Scenario: Exporting data from a Riak bucket to a local file* on page 3255.
tRiakInput

Extracts the desired data from a bucket in a Riak node so as to store or apply changes to the data.
tRiakInput reads data from a Riak bucket and send data in the Talend flow.

**tRiakInput Standard properties**

These properties are used to configure tRiakInput running in the Standard Job framework.
The Standard tRiakInput component belongs to the Big Data and the Databases families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit Schema</th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• View schema: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
<td></td>
</tr>
<tr>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td>Use existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. If you select the Use an existing connection check box, the Nodes table will not be available.</td>
</tr>
<tr>
<td>Nodes</td>
<td>Fill in the information required for a connection to a Riak cluster.</td>
</tr>
<tr>
<td>• Host: enter the IP address or hostname of a Riak node in the Riak cluster that you want to connect to.</td>
<td></td>
</tr>
<tr>
<td>• Port: enter the listening port number of a Riak node. You can leave this field empty to use its default value 8098.</td>
<td></td>
</tr>
<tr>
<td>• Riak path: enter the Riak path for accessing a Riak node. You can leave this field empty to use its default value riak.</td>
<td></td>
</tr>
</tbody>
</table>
One example of the URL for accessing a Riak node is http://127.0.0.1:8098/riak.

For more information about the concepts related to Riak, see http://docs.basho.com/riak/latest/theory/concepts/.

<table>
<thead>
<tr>
<th>Bucket</th>
<th>Type in the name of the bucket from which you want to read data.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Type in the key which is associated with the data that you want to read.</td>
</tr>
<tr>
<td>Output key to column</td>
<td>Select this check box and from the list select the desired column to which the keys will be output.</td>
</tr>
<tr>
<td>Values column</td>
<td>Customize the columns to which the values will be output.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>

**Usage**

| Usage rule | This component always needs an output link. |

**Scenario: Exporting data from a Riak bucket to a local file**

This scenario applies only to a Talend solution with Big Data.
The following scenario describes a Job which reads data from a Riak bucket and writes it into a local `.txt` file.

**Prerequisites:** The Riak bucket from which you want to export data already exists. In this example, the data from the bucket *computer* will be exported and the bucket has already imported the following data:

```
id; company; brand; price; owner
001; Dell; Inspiron 15; 299; Amanda
002; Dell; Inspiron 15R; 549; Linda
003; HP; Pavilion 500-210qe; 539; Marina
004; HP; Pavilion 500-075; 599; Diana
```

**Dropping and linking components**

**Procedure**

1. Drop the following components from the **Palette** to the design workspace: `tRiakConnection`, `tRiakInput`, `tFileOutputDelimited`, and `tRiakClose`.
2. Connect `tRiakConnection` to `tRiakInput` using a **Trigger > On Subjob Ok** link.
3. Connect `tRiakInput` to `tFileOutputDelimited` using a **Row > Main** link.
4. Connect `tFileOutputDelimited` to `tRiakClose` using a **Trigger > On Component Ok** link.

**Configuring the components**

**Opening a connection to Riak**

**Procedure**

1. Double-click `tRiakConnection` to open its **Basic settings** view in the **Component** tab.

   ![tRiakConnection Basic Settings](image)

   **Connection Nodes**

   2. In the **Nodes** table, enter the information of a Riak cluster you want to connect to.
Exporting data from a Riak bucket to a local file

Procedure

1. Double-click \texttt{tRiakInput} to open its Basic settings view in the Component tab.

2. Click Edit schema to define the structure of exported data. In this example, three columns are defined: \texttt{id}, \texttt{company}, and \texttt{price}.

3. Select the \textbf{Use an existing connection} check box and then select the connection you have configured earlier. In this example, it is \texttt{tRiakConnection_1}.

4. In the Bucket field, enter the name of the bucket from which the data will be exported, \texttt{computer} in this example.
5. Select the **Output key to column** check box, and select the desired column from the list. *id* is selected in this example.

6. In the **Value columns** table, click twice the button and select *company* and *price* from the list respectively.

**Writing data into a local file**

**Procedure**

1. Double-click **tFileOutputDelimited** to open its **Basic settings** view in the **Component** tab.

2. In the **File Name** field, enter the full path to the local file in which you want to store the exported data, *D:/Output/computer.txt* in this example.

3. Select the **Include Header** check box.

4. Leave other settings as they are.

**Closing the connection to Riak**

**Procedure**

1. Double-click **tRiakClose** to open its **Basic settings** view in the **Component** tab.

2. Select the connection you want to close from the **Component List**, **tRiakConnection_1** in this example.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.

2. Execute the Job by pressing **F6** or clicking **Run** on the **Run** tab.

3. Go to the local directory where the file is stored and check the exported data from the Riak bucket.
<table>
<thead>
<tr>
<th>id</th>
<th>company</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>HP</td>
<td>539</td>
</tr>
<tr>
<td>004</td>
<td>HP</td>
<td>599</td>
</tr>
<tr>
<td>001</td>
<td>Dell</td>
<td>299</td>
</tr>
<tr>
<td>002</td>
<td>Dell</td>
<td>549</td>
</tr>
</tbody>
</table>
tRiakKeyList

Retrieves a list of keys and iterates on it within a Riak bucket for analysis or development purposes.

tRiakKeyList Standard properties

These properties are used to configure tRiakKeyList running in the Standard Job framework.

The Standard tRiakKeyList component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. If you select the Use an existing connection check box, the Nodes table will not be available.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>Fill in the information required for a connection to a Riak cluster.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Host</strong>: enter the IP address or hostname of a Riak node in the Riak cluster that you want to connect to.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Port</strong>: enter the listening port number of a Riak node. You can leave this field empty to use its default value 8098.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Riak path</strong>: enter the Riak path for accessing a Riak node. You can leave this field empty to use its default value riak.</td>
</tr>
<tr>
<td></td>
<td>One example of the URL for accessing a Riak node is <a href="http://127.0.0.1:8098/riak">http://127.0.0.1:8098/riak</a>.</td>
</tr>
<tr>
<td></td>
<td>For more information about the concepts related to Riak, see <a href="http://docs.basho.com/riak/latest/theory/concepts/">http://docs.basho.com/riak/latest/theory/concepts/</a>.</td>
</tr>
<tr>
<td>Bucket</td>
<td>Type in the name of the bucket from which you want to retrieve all keys.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStatCatcher Statistics    | Select this check box to gather the Job processing metadata at the Job level as well as at each component level.                                                                                                                                                |

Global Variables

| Global Variables           | **CURRENT_KEY**: the current key. This is a Flow variable and it returns a string.                                                                                                  |
**tRiakKeyList**

**NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | This component can be used as a standalone component. |

### Related scenarios

No scenario is available for the Standard version of this component yet.
tRiakOutput

Receives data from the preceding component and writes data into or deletes data from a bucket in a Riak cluster.

**tRiakOutput Standard properties**

These properties are used to configure tRiakOutput running in the Standard Job framework.

The Standard tRiakOutput component belongs to the Big Data and the Databases families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:
| | • **View schema**: choose this option to view the schema only.
| | • **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
| | • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.
| **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
| **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.
| When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.
<p>| You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>). |</p>
<table>
<thead>
<tr>
<th><strong>Use existing connection</strong></th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined. If you select the <strong>Use an existing connection</strong> check box, the <strong>Nodes</strong> table will not be available.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nodes</strong></td>
<td>Fill in the information required for a connection to a Riak cluster. <strong>Host</strong>: enter the IP address or hostname of a Riak node in the Riak cluster that you want to connect to. <strong>Port</strong>: enter the listening port number of a Riak node. You can leave this field empty to use its default value 8098. <strong>Riak path</strong>: enter the Riak path for accessing a Riak node. You can leave this field empty to use its default value <code>riak</code>. One example of the URL for accessing a Riak node is <code>http://127.0.0.1:8098/riak</code>. For more information about the concepts related to Riak, see <a href="http://docs.basho.com/riak/latest/theory/concepts/">http://docs.basho.com/riak/latest/theory/concepts/</a>.</td>
</tr>
<tr>
<td><strong>Bucket</strong></td>
<td>Specify the name of the bucket to which you want to apply changes.</td>
</tr>
<tr>
<td><strong>Action on data</strong></td>
<td>In the specified bucket, you can perform: <strong>Upsert</strong>: Insert data if they do not exist or update the existing data. <strong>Delete</strong>: Remove values associated with the specified key. If you select <strong>Delete</strong> from the <strong>Action on data</strong> list, only the <strong>Key column</strong> list is available in the <strong>Key</strong> area.</td>
</tr>
<tr>
<td><strong>Auto generate the key</strong></td>
<td>Select this check box to let the Riak system generate keys for the values automatically.</td>
</tr>
<tr>
<td><strong>Key column</strong></td>
<td>Select one column from the list to write its data into the Riak bucket as keys. Note that the key must be unique across one bucket.</td>
</tr>
<tr>
<td><strong>Value columns</strong></td>
<td>Customize the columns to write their data into the Riak bucket as values.</td>
</tr>
<tr>
<td><strong>Advanced settings</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>
### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>UPSERT_NB_LINE: the number of rows inserted and updated. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DELETE_NB_LINE: the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

| Usage rule | This component is used as an output component and it always needs an incoming link. |

### Related scenarios

No scenario is available for the Standard version of this component yet.
**tRouteFault**

Sends messages from a Data Integration Job to a Mediation Route and mark the message as fault.

**tRouteFault Standard properties**

These properties are used to configure tRouteFault running in the Standard Job framework.

The Standard tRouteFault component belongs to the ESB family.

*This component is relevant only when used with one of the *Talend* solutions with ESB, as it should be used with the Service Repository node and the Data Service creation related wizard(s).*

### Basic settings

| **Output Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  • **View schema**: choose this option to view the schema only.  
  • **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
  • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.  
Click Sync columns to retrieve the schema from the previous component connected in the Job. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the Repository file where Properties are stored.</td>
<td></td>
</tr>
<tr>
<td><strong>Camel</strong></td>
<td>Enter the message parameters corresponding to the columns you defined in the schema dialog box via the Edit schema button.</td>
</tr>
</tbody>
</table>
| **Type** | Select between:  
  • **Body**: Used to set the body of a Route, equals the code exchange.getOut().setBody(...).  
  • **Property**: Used to set the property of a Route, equals the code exchange.setProperty(name, value).  
  • **System**: Used to set the system property, equals the code System.setProperty(name, value). |
### Header

Used the set the output header, equals the code `exchange.getOut().setHeader(name, value)`.

| Name | This column is used to set the name for the parameter when its type is either Property, System or Header. This column is ignored when the type is Body. |

### Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

### Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

### Usage

| Usage rule | This component can be used as the end component and thus requires an input component. |

| Limitation | The Job using a tRouteFault must be launched only within a Route using the cTalendJob component. |

### Scenario: Exchanging messages between a Job and a Route

This scenario applies only to a Talend solution with ESB.

This scenario describes how to exchange messages between a Job and a Route using the tRouteInput and the tRouteFault component. To do this, a Job is created to receive the message from the Route and mark the message as fault. Then a Route is built to send the message to the Job and handle the fault message.

### Creating an Data Integration Job

In this section, a Job named RouteCommunication_tRouteFault is built.
Arranging the flow of the message

About this task

1. Drag and drop a tRouteInput and a tRouteFault from the Palette onto the design workspace.
2. Right-click the tRouteInput component, select Row > Main from the contextual menu and click the tRouteFault component.

Configuring how the message is processed

1. Double-click the tRouteInput component to open its Basic settings view in the Component tab.
2. Click the three-dot button next to Edit Schema. In the schema dialog box, click the plus button to add a new line of String type and name it body. Click OK to close the dialog box.
3. In the **Simple Expression** field for the *body* element, enter "\${in.body}" to get the body of the input message from the Route.

4. Double-click the **tRouteFault** component to display its **Basic settings** view in the **Component** tab.

5. Make sure that **tRouteFault** has one element *body*. Set its type to **Body**.

6. Press **Ctrl+S** to save your Job.

### Creating a Mediation Route

In this section, we will create a Route to send a message to the Job and handle the fault message.

#### Dropping and linking the components

**About this task**

**Procedure**

1. Drag and drop a **cTimer**, a **cSetBody**, a **cJavaDSLProcessor**, a **cTalendJob**, a **cOnException** and a **cProcessor** component from the **Palette** onto the design workspace.

2. Link the components with the **Row > Route** connection as shown above.

3. Label the components for better identification of their roles.

**Configuring the components**

**Procedure**

1. Double-click the **cTimer** component to open its **Basic settings** view in the **Component** tab.
2. Enter 1 in the **Repeat** field to trigger a message exchange one time.

3. Double-click the **cSetBody** component to display its **Basic settings** view in the **Component** tab.

4. Select **Simple** from the **Language** list and enter *Hello!* in the **Expression** field.

5. Double-click the **cJavaDSLProcessor** component to open its **Basic settings** view in the **Component** tab.

6. Enter `.handleFault()` in the **Code** box so that the fault message will be caught as an exception.

7. Double-click the **cTalendJob** component to display its **Basic settings** view in the **Component** tab.
8. Select Job `RouteCommunication_tRouteFault` from the repository to call the Job that is created in the previous section.

9. Double click the `cOnException` component to display its Basic settings view in the Component tab.

10. Click [+] to add a line to the Exceptions table, and enter `org.apache.camel.CamelException` in the Exception field to handle the exceptions.

11. Double-click the `cProcessor` component to display its Basic settings view in the Component tab.
12. Customize the code in the Code area to show the message body:

   ```java
   System.out.println(exchange.getIn());
   ```

13. Press Ctrl+S to save your Route.

**Executing the Route**

Click the Run view to display it and click the Run button to launch the execution of your Route. You can also press F6 to execute it.

The message *Hello!* is sent from the Route to the Job and caught as exception:
[statistics] connecting to socket on port 3783
[statistics] connected

 Processed by failure processor:
 FallbackErrorHandler[Channel[DelegateSync[local_project.routefault_0_1.tRouteFault@102511045b]]

Message History

<table>
<thead>
<tr>
<th>RouteId</th>
<th>ProcessorId</th>
<th>Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elapsed (ms)</td>
<td>[Starter_cTimer_1]</td>
<td>[Starter_cTimer_1]</td>
</tr>
<tr>
<td>[timer://cTimer_1?delay=1000&amp;repeatCount=1]</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>[Starter_cTimer_1]</td>
<td>[cJavaDSLProcessor_]</td>
<td>[setBody{simple{Hello!}}]</td>
</tr>
<tr>
<td>[20]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Starter_cTimer_1]</td>
<td>[cTalendJob_1]</td>
<td></td>
</tr>
<tr>
<td>[talend://local_project.routecommunication_routefault_0_1.RouteCommunication_t]</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>[cProcessor_1]</td>
<td>[Processor@0x251104fb]</td>
<td></td>
</tr>
<tr>
<td>[0]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Exchange

<table>
<thead>
<tr>
<th>Exchange</th>
<th>Id</th>
<th>ExchangePattern</th>
<th>Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>[cRgCamelException. Hello!]</td>
<td>ID-Talend-PC-7919-1437988941394-0-2</td>
<td>InOnly</td>
<td></td>
</tr>
<tr>
<td>Headers</td>
<td>{breed:crumbId-ID-Talend-PC-7919-1437988941394-0-1, CamelRedelivered=false, CamelRedeliveryCounter=0, firedTime=Mon Jul 27 17:22:23 CST 2015}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BodyType</td>
<td>String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Body</td>
<td>Hello!</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stacktrace

org.apache.camel.CamelException: Hello!
at org.apache.camel.processor.interceptor.HandleFaultInterceptor.handleFault(HandleFaultInterceptor.java:75)
tRouteInput

Accepts messages in a Data Integration Job from a Mediation Route.

tRouteInput Standard properties

These properties are used to configure tRouteInput running in the Standard Job framework.

The Standard tRouteInput component belongs to the ESB family.

This component is relevant only when used with one of the Talend solutions with ESB, as it should be used with the Service Repository node and the Data Service creation related wizard(s).

Basic settings

| Input Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:
| • View schema: choose this option to view the schema only.
| • Change to built-in property: choose this option to change the schema to Built-in for local changes.
| • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |

| Built-in: No property data stored centrally. |

| Repository: Select the Repository file where Properties are stored. |

| Camel | Enter the message parameters corresponding to the columns you defined in the schema dialog box via the Edit schema button using the Simple Expression Language. For more information about the Simple Expression Language, see the site http://camel.apache.org/simple.html. |

Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |
Global Variables

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*. |

Usage

| Usage rule | This component can be used as starter component and has an output. It is mandatory that a **tRouteInput** component is used in the Data Integration Job to be called by a Route using the **cTalendJob** component. The reason for it is that this will prevent the referenced Job from starting automatically when deployed in Talend Runtime. Instead it will only start when it gets called by the Route.

For more information about how to use the **cTalendJob** component, see Talend Help Center (https://help.talend.com). |

| Limitation | The Job using a **tRouteInput** must be launched only within a Route using the **cTalendJob** component. |

Scenario: Exchanging messages between a Job and a Route

This scenario applies only to a Talend solution with ESB.

This scenario describes how to accept messages from a Route using the **tRouteInput** component and send messages to a Route using the **tRouteOutput** component. To do this, we need to build a Job and a Route to call the Job using the **cTalendJob** component.

For more information about how to use the **cTalendJob** component, see Talend Help Center (https://help.talend.com).

Creating an Data Integration Job

In this section, we will build a Job named *RouteCommunication* to accept the message from a Route, change the message body and send the message back to the Route.
Arranging the flow of the message

About this task

Procedure

1. Drag and drop a tRouteInput, a tLogRow, a tReplace and a tRouteOutput from the Palette onto the design workspace.
2. Link the components using the Row > Main connection as shown above.

Configuring how the message is processed

Procedure

1. Double-click the tRouteInput component to open its Basic settings view in the Component tab.

2. Click the three-dot button next to Edit Schema. In the schema dialog box, click the plus button to add a new line of String type and name it body. Click OK to close the dialog box.

3. In the Simple Expression field for the body element, enter "${in.body}" to get the body of the input message from the Route.
4. The **tLogRow** component will monitor the message exchanges and does not need any configuration.

5. Double-click the **tReplace** component to open its **Basic settings** view in the **Component** tab.

6. Click **Sync columns** to retrieve the schema from the **tLogRow**. In the pop-up dialog box, click **Yes** to propagate the schema to the subsequent component.

   Click [+] to add a row to the **Search/Replace** table. Enter *World* in the **Search** column and *Talend* in the **Replace with** column. Clear the **Whole word** check box.

7. Double-click the **tRouteOutput** component to display its **Basic settings** view in the **Component** tab.

8. Make sure that **tRouteOutput** has one element **body**. Set its type to **Body**.

9. Press **Ctrl+S** to save your Job.

**Creating a Mediation Route**

In this section, we will create a Route to send a message to the Job and receive the message sent back to the Route.
Dropping and linking the components

About this task

Procedure

1. Drag and drop a cTimer, a cSetBody, a cTalendJob, and a cLog from the Palette onto the design workspace.

2. Link the components with the Row > Route connection as shown above.

3. Label the components for better identification of their roles.

Configuring the components

Procedure

1. Double-click the cTimer component to open its Basic settings view in the Component tab.

2. In the Repeat field, enter 1 to generate the message exchange one time. Keep the default settings of the other options.

3. Double-click the cSetBody component to display its Basic settings view in the Component tab.

4. Select Simple from the Language list and enter Hello World! in the Expression field.

5. Double-click the cTalendJob component to display its Basic settings view in the Component tab.
6. Select Job *RouteCommunication* from the repository.
7. The *cLog* component will monitor the message exchanges and does not need any configuration.
8. Press **Ctrl+S** to save your Route.

**Executing the Route**

Click the **Run** view to display it and click the **Run** button to launch the execution of your Route. You can also press **F6** to execute it. In the execution log you can see that the message *Hello World!* is sent to the Job. The message body is changed to *Hello Talend!* and sent back to the Route.
tRouteOutput

Sends messages from a Data Integration Job to a Mediation Route.

**tRouteOutput Standard properties**

These properties are used to configure tRouteOutput running in the Standard Job framework.

The Standard tRouteOutput component belongs to the ESB family.

*This component is relevant only when used with one of the Talend solutions with ESB, as it should be used with the Service Repository node and the Data Service creation related wizard(s).*

**Basic settings**

| Output Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  • View schema: choose this option to view the schema only.  
  • Change to built-in property: choose this option to change the schema to Built-in for local changes.  
  • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.  
  
  Click Sync columns to retrieve the schema from the previous component connected in the Job. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the Repository file where Properties are stored.</td>
<td></td>
</tr>
<tr>
<td>Camel</td>
<td>Enter the message parameters corresponding to the columns you defined in the schema dialog box via the Edit schema button.</td>
</tr>
</tbody>
</table>
| Type | Select between:  
  • **Body**: Used to set the body of a Route, equals the code exchange.getOut().setBody(...).  
  • **Property**: Used to set the property of a Route, equals the code exchange.setProperty(name, value).  
  • **System**: Used to set the system property, equals the code System.setProperty(name, value). |
Header: Used the set the output header, equals the code exchange.getOut().setHeader(name, value).

Name
This column is used to set the name for the parameter when its type is either Property, System or Header. This column is ignored when the type is Body.

Advanced settings

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

Global Variables

**Global Variables**
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

**Usage rule**
This component can be used as the end component and thus requires an input component. It is mandatory that a tRouteInput component is used in the Data Integration Job to be called by a Route using the cTalendJob component. The reason for it is that this will prevent the referenced Job from starting automatically when deployed in Talend Runtime. Instead it will only start when it gets called by the Route.

For more information about how to use the cTalendJob component, see Talend Help Center (https://help.talend.com).

**Limitation**
The Job using a tRouteOutput must be launched only within a Route using the cTalendJob component.

Related scenario

For a scenario in which tRouteOutput is used, see Scenario: Exchanging messages between a Job and a Route on page 3274.
tRowGenerator

Creates an input flow in a Job for testing purposes, in particular for boundary test sets.
tRowGenerator generates as many rows and fields as are required using random values taken from a list.

tRowGenerator Standard properties

These properties are used to configure tRowGenerator running in the Standard Job framework.
The Standard tRowGenerator component belongs to the Misc family.
The component in this framework is generally available.

Basic settings

| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:   
• View schema: choose this option to view the schema only.  
• Change to built-in property: choose this option to change the schema to Built-in for local changes.  
• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>
| RowGenerator editor | The editor allows you to define the columns and the nature of data to be generated. You can use predefined routines or type in the function to be used to generate the data specified. 
Note that in a Storm Job, the value -1 in the Number of rows for RowGenerator field in the RowGenerator editor means to generate infinite rows of input data. |
Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

Global Variables

| Global Variables | NB.LINE: the number of rows processed. This is an After variable and it returns an integer. |
|                 | ERROR.MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |
|                 | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
|                 | To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. |
|                 | For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | The tRowGenerator Editor’s ease of use allows users without any Java knowledge to generate random data for test purposes. |

Defining the schema

First you need to define the structure of data to be generated.

- Add as many columns to your schema as needed, using the plus (+) button.
- Type in the names of the columns to be created in the Columns area and select the Key check box if required.
- Make sure you define then the nature of the data contained in the column, by selecting the Type in the list. According to the type you select, the list of Functions offered will differ. This information is therefore compulsory.

<table>
<thead>
<tr>
<th>Column</th>
<th>Key</th>
<th>Type</th>
<th>Nullable</th>
<th>Functions</th>
<th>Preview</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_employees</td>
<td></td>
<td>int</td>
<td></td>
<td>sequence</td>
<td>2</td>
</tr>
<tr>
<td>First_Name</td>
<td></td>
<td>String</td>
<td></td>
<td>...</td>
<td>Phoebe</td>
</tr>
<tr>
<td>Last_Name</td>
<td></td>
<td>String</td>
<td></td>
<td>getUserName</td>
<td>Eisenhower</td>
</tr>
<tr>
<td>Hire_Date</td>
<td></td>
<td>Day</td>
<td></td>
<td>getRandomDate</td>
<td>2008-08-31</td>
</tr>
</tbody>
</table>
• Some extra information, although not required, might be useful such as Length, Precision or Comment. You can also hide these columns, by clicking on the Columns drop-down button next to the toolbar, and unchecking the relevant entries on the list.

• In the Function area, you can select the predefined routine/function if one of them corresponds to your needs. You can also add to this list any routine you stored in the Routine area of the Repository. Or you can type in the function you want to use in the Function definition panel.

Related topic: see Talend Studio User Guide.

• Click Refresh to have a preview of the data generated.

• Type in a number of rows to be generated. The more rows to be generated, the longer it’ll take to carry out the generation operation.

Defining the function

Select the […] under Function in the Schema definition panel in order to customize the function parameters.

• Select the Function parameters tab

• The Parameter area displays Customized parameter as function name (read-only)

• In the Value area, type in the Java function to be used to generate the data specified.

• Click on the Preview tab and click Preview to check out a sample of the data generated.

Scenario: Generating random java data

The following scenario creates a two-component Job, generating 50 rows structured as follows: a randomly picked-up ID in a 1-to-3 range, a random ascii First Name and Last Name generation and a random date taken in a defined range.

• Drop a tRowGenerator and a tLogRow component from the Palette to the design workspace.

• Right-click tRowGenerator and select Row > Main. Drag this main row link onto the tLogRow component and release when the plug symbol displays.

• Double click tRowGenerator to open the Editor.

• Define the fields to be generated.
• The random ID column is of integer type, the First and Last names are of string type and the Date is of date type.

• In the **Function** list, select the relevant function or set on the three dots for custom function.

• On the **Function parameters** tab, define the Values to be randomly picked up.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>customize parameter</td>
<td>1,2,3</td>
<td></td>
</tr>
</tbody>
</table>

• **First_Name** and **Last_Name** columns are to be generated using the `getAsciiRandomString` function that is predefined in the system routines. By default the length defined is 6 characters long. You can change this if need be.

• The **Date** column calls the predefined `getRandomDate` function. You can edit the parameter values in the **Function parameters** tab.

• Set the **Number of Rows** to be generated to 50.

• Click **OK** to validate the setting.

• Double click **tLogRow** to view the Basic settings. The default setting is retained for this Job.

• Press **F6** to run the Job.

---

Running process with context: Default.

---

...
The 50 rows are generated following the setting defined in the `tRowGenerator` editor and the output is displayed in the Run console.
**tRSSInput**

Reads RSS-Feeds using URLs. tRSSInput makes it possible to keep track of blog entries on websites to gather and organize information for quick and easy access.

**tRSSInput Standard properties**

These properties are used to configure tRSSInput running in the Standard Job framework.

The Standard tRSSInput component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Schema and Edit Schema</strong></th>
<th>A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The tRSSInput component has a read-only schema that is made of four columns: TITLE, DESCRIPTION, PUBDATE, and Link.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RSS URL</strong></td>
<td>Enter the URL for the RSS_Feed to read.</td>
</tr>
<tr>
<td><strong>Read articles from</strong></td>
<td>If selected, tRSSInput reads articles on the RSS_Feed from the date set through the three-dot [...] button next to the date time field.</td>
</tr>
<tr>
<td><strong>Max number of articles</strong></td>
<td>If selected, tRSSInput reads as many articles as the number entered in the max amount field.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Global Variables**

| **Global Variables** | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
|----------------------|---------------------------------------------------------------------------------------------------|
|                      | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
|                      | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
|                      | To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. |
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is generally used as an input component. It requires an output component.</th>
</tr>
</thead>
</table>

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |

Scenario: Fetching frequently updated blog entries.

This two-component scenario aims at retrieving frequently updated blog entries from a Talend local news RSS feed using the tRSSInput component.

Procedure

Procedure

1. Drop the following components from the Palette onto the design workspace: tRSSInput and tLogRow.
2. Right-click to connect them using a Row > Main link.
3. In the design workspace, select tRSSInput, and click the Component tab to define the basic settings for tRSSInput.

4. Enter the URL for the RSS_Feed to access. In this scenario, tRSSInput links to the Talend RSS_Feed: http://feeds.feedburner.com/Talend.
5. Select/clear the other check boxes as required. In this scenario, we want to display the information about two articles dated from July 20, 2008.

6. In the design workspace, select tLogRow and click the Component tab to define its basic settings. For more information about tLogRow properties, see tLogRow Standard properties on page 1988.

7. Save the Job and press F6 to execute it.

The tRSSInput component accessed the RSS feed of Talend website on your behalf and organized the information for you.

Two blog entries are displayed on the console. Each entry has its own title, description, publication date, and the corresponding RSS feed URL address. Blogs show the last entry first, and you can scroll down to read earlier entries.
tRSSOutput

Creates and writes XML files that hold RSS or Atom feeds.

### tRSSOutput Standard properties

These properties are used to configure tRSSOutput running in the Standard Job framework.

The Standard tRSSOutput component belongs to the Internet family.

The component in this framework is generally available.

#### Basic settings

<table>
<thead>
<tr>
<th>File name</th>
<th>Name or path to the XML file to be processed and/or the variable to be used. For further information about how to define and use a variable in a Job, see <a href="#">Talend Studio User Guide</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding</td>
<td>Select an encoding type from the list, or select Custom and define it manually. This field is compulsory for DB data handling.</td>
</tr>
<tr>
<td>Append</td>
<td>Select this check box to add the new rows to the end of the file.</td>
</tr>
<tr>
<td>Mode</td>
<td>Select between RSS or ATOM according to the feed you want to generate.</td>
</tr>
</tbody>
</table>

**Channel (in RSS mode)**

- **Title**: Enter a meaningful title.
- **Description**: Enter a description that you think will describe your content.
- **Publication date**: Enter the relevant date.
- **Link**: Enter the relevant URL.

**Note:**

The information to be typed in here concerns your entire input data, site etc, rather than a particular item.

**Feed (in ATOM mode)**

- **Title**: Enter a meaningful title.
- **Link**: Enter the relevant URL.
- **Id**: Enter the valid URL corresponding to the Link.
- **Update date**: Enter the relevant date.
- **Author name**: Enter the relevant name.

**Optionnal Channel Elements**

Click the [+I button below the table to add new lines and enter the information relative to the RSS flow metadata:

- **Element Name**: name of the metadata.
- **Element Value**: content of the metadata.
### Schema and Edit Schema

A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

By default, the schema of **tRSSOutput** is made of five read-only columns: **id**, **title**, **link**, **updated**, and **summary**. You can add new columns or click **Syn columns** to retrieve the schema structure from the preceding component.

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</th>
</tr>
</thead>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
</tbody>
</table>
Usage

| Usage rule | This component must be linked to an input or intermediary component. |

Scenario 1: Creating an RSS flow and storing files on an FTP server

In this scenario we:

- create an RSS flow for files that you would like to share with other people, and
- store the complete files on an FTP server.

This scenario writes an RSS feed XML file about a Mysql table holding information about books. It adds links to the files stored on an FTP server in case users want to have access to the complete files.

Dropping and linking components

Procedure

1. Drop the following components from the Palette onto the design workspace: tMysqlInput, tRSSOutput, and tFTPPut.
2. Right-click tMysqlInput and connect it to tRSSOutput using a Row > Main link.
3. Right-click tMysqlInput and connect it to tFTPPut using a Trigger > OnSubjobOk link.

Defining the data source

Procedure

1. In the design workspace, select tMysqlInput, and click the Component tab to define the basic settings for tMysqlInput.
2. Set the **Property type** to **Repository** and click the three-dots button [...] to select the relevant DB entry from the list. The connection details along with the schema get filled in automatically.

3. In the **Table Name** field, either type your table name or click the three dots button [...] and select your table name from the list. In this scenario, the Mysql input table is called "rss_talend" and the schema is made up of four columns, **TITLE**, **Description**, **PUBDATE**, and **LINK**.

4. In the **Query** field, enter your DB query paying particular attention to properly sequence the fields in order to match the schema definition, or click **Guess Query**.

**Creating an RSS flow**

**Procedure**

1. In the design workspace, select **tRSSOutput**, and click the **Component** view to define the basic settings for **tRSSOutput**.
2. In the File name field, use the by default file name and path, or browse to set your own for the output XML file.

3. Select the encoding type on the Encoding Type list.

4. In the Mode area, select RSS.

5. In the Channel panel, enter a title, a description, a publication date, and a link to define your input data as a whole.

6. Click Edit Schema to modify the schema if necessary.

   **Note:**
   
   You can click Sync columns to retrieve the generated schema from the preceding component.

7. Save your Job and press F6 to execute this first part.
The **tRSSOutput** component created an output RSS flow in an XML format for the defined files.

**Writing the complete files to an FTP server**

**About this task**

To store the complete files on an FTP server:

**Procedure**

1. In the design workspace, select **FTPPut**, and click the **Component** tab to define the basic settings for **tFTPPut**.
2. Enter the host name and the port number in their corresponding fields.
3. Enter your connection details in the corresponding **Username** and **Password** fields.
4. Browse to the local directory, or enter it manually in the **Local directory** field.
5. Enter the details of the remote server directory.
6. Select the transfer mode from the **Transfer mode** list.
7. On the **Files** panel, click on the plus button to add new lines and fill in the filemasks of all files to be copied onto the remote directory. In this scenario, the files to be saved on the FTP server are all text files.
8. Save your Job and press **F6** to execute it.

   Files defined in the Filemask are copied on the remote server.

### Scenario 2: Creating an RSS flow that contains metadata

This scenario describes a two-component Job that creates an RSS flow that holds metadata and then redirects the obtained information in an XML file of the output RSS flow.

### Dropping and linking components

**Procedure**

1. Drop **tRSSInput** and **tRSSOutput** from the **Palette** to the design workspace.
2. Connect the two components together using a **Row > Main** link.

### Configuring the components

**Procedure**

1. Double-click **tRSSInput** to open its **Basic settings** view and define the component properties.
2. Enter the URL for the RSS Feed to access. In this scenario, tRSSInput links to the Talend RSS_Feed: http://feeds.feedburner.com/Talend.

3. In the design workspace, double-click tRSSOutput to display its Basic settings view and define the component properties.

4. In the File name field, use the by default file name and path, or browse to set your own for the output XML file.

5. Select the encoding type on the Encoding Type list.

6. In the Mode area, select RSS.

7. In the Channel panel, enter a title, a description, a publication date and a link to define your input data as a whole.

8. In the Optional Channel Element, define the RSS flow metadata. In this example, the flow has two metadata: copyright, which value is tos, and language which value is en_us.

9. Click Edit Schema to modify the schema if necessary.

Note:

You can click Sync Column to retrieve the generated schema from the preceding component.
Saving and executing the Job

Procedure

1. Press Ctrl+S to save your Job.
2. Press F6 or click Run on the Run tab to execute the Job.

![Output XML file](image)

The defined files are copied in the output XML file and the metadata display under the `<channel>` node above the information about the RSS flow.

Scenario 3: Creating an ATOM feed XML file

This scenario describes a two component Job that generates data and writes them in an ATOM feed XML file.

Dropping and linking components

Procedure

1. Drop the following components from the Palette onto the design workspace: tFixedFlowInput of the Misc component group and tRSSOutput of the Internet component group.
2. Right-click tFixedFlowInput and connect it to tRSSOutput using a Row Main link.
3. When asked whether you want to pass on the schema of tRSSOutput to tFixedFlowInput, click Yes.
Configuring the components

Procedure

1. In the design workspace, double-click tFixedFlowInput to display its corresponding Component view and define its basic settings.

![tFixedFlowInput_1](image)

- **Basic settings**
  - **Schema**: Built-In
  - **Number of rows**: 1
- **Dynamic settings**
  - **Mode**: Use Single Table
  - **Values**: ID "urn:uuid:1225c6b95-cdb8-4abb-aaaa-80da344efa69", Title "Job using tRSSOutput in Talend Open Studio", Link "http://feeds.feedburner.com/Talend")", Updated "2010-09-07T15:43:02Z", Summary "This is a Job made in Talend Open Studio and usn..."

2. In the **Number of rows** field, leave the default setting to 1 to only generate one line of data.
3. In the **Mode** area, leave the **Use Single Table** option selected and fill in the **Values** table. Note that the **Column** field of the **Values** table is filled in by the columns of the schema defined in the component.
4. In the **Value** field of the **Values** table, type in the data you want to be sent to the following component.
5. In the design workspace, double-click tRSSOutput to display its corresponding Component view and define its basic settings.
6. Click the [...] button next to the File Name field to set the output XML file directory and name.

7. In the Mode area, select ATOM to generate an ATOM feed XML file.

   **Warning:**
   
   As the ATOM feed format is strict, some default information is required to create the XML file. So, the schema tRSSOutput contains default columns that will contain those information. Those default columns are greyed out to indicate that they must not be modified. If you choose to modify the schema of the component, the ATOM XML file created will not be valid.

8. In the Feed area, enter a title, link, id, update date, author name to define your input data as a whole.

**Saving and executing the Job**

**Procedure**

1. Press Ctrl+S to save your Job.
2. Press F6 or click Run on the Run tab to execute the Job.
The `tRSSOutput` component creates an output ATOM flow in an XML format.
**tRunJob**

Manages complex Job systems which need to execute one Job after another.

tRunJob executes the Job called in the component’s properties, in the frame of the context defined.

**tRunJob Standard properties**

These properties are used to configure tRunJob running in the Standard Job framework.

The Standard tRunJob component belongs to the System and the Orchestration families.

The component in this framework is generally available.

⚠️ **Warning:**

The tRunJob component is supported with limitations, which means that only S4 (Minor) support cases are accepted and no patches are provided. If you use tRunJob within Data Services and Routes (with cTalendJob), support is provided on a "best effort" basis only. In most cases, there are class loading issues which can sometimes be resolved but not always.

This is because tRunJob is not designed to work in a Service/Route style (ESB) deployment, so regular support is not provided if you decide to use it, even though it may work in many cases. If you used tRunJob in the past, it is recommended to change your Job Design to use Joblets instead.

For DI and non-ESB use cases, it is still a valuable component and has support.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

| **Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**. |
Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Copy Child Job Schema</th>
<th>Click to fetch the child Job schema.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use dynamic job</strong></td>
<td>Select this check box to allow multiple Jobs to be called and processed. When this option is enabled, only the latest version of the Jobs can be called and processed. An independent process will be used to run the subjob. The Context and the Use an independent process to run subjob options disappear.</td>
</tr>
<tr>
<td><strong>Warning:</strong> The Use dynamic job option is not compatible with the Jobserver cache. Therefore, the execution may fail if you run a Job that contains tRunjob with this check box selected in Talend Administration Center.</td>
<td></td>
</tr>
<tr>
<td><strong>Warning:</strong> This option is incompatible with the Use or register a shared DB Connection option of database connection components. When tRunJob works together with a database connection component, enabling both options will cause your Job to fail.</td>
<td></td>
</tr>
<tr>
<td><strong>Warning:</strong> This option is not supported within ESB Routes or Data Services.</td>
<td></td>
</tr>
<tr>
<td><strong>Context job</strong></td>
<td>This field is visible only when the Use dynamic job option is selected. Enter the name of the Job that you want to call from the list of Jobs selected.</td>
</tr>
<tr>
<td><strong>Job</strong></td>
<td>Select the Job to be called in and processed. Make sure you already executed once the Job called, beforehand, in order to ensure a smooth run through tRunJob.</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>Select the child Job version that you want to use.</td>
</tr>
<tr>
<td><strong>Context</strong></td>
<td>If you defined contexts and variables for the Job to be run by the tRunJob, select the applicable context entry on the list.</td>
</tr>
<tr>
<td><strong>Use an independent process to run subjob</strong></td>
<td>Select this check box to use an independent process to run the subjob. This helps in solving issues related to memory limits.</td>
</tr>
<tr>
<td><strong>Warning:</strong> This option is not compatible with the Jobserver cache. Therefore, the execution may fail if you run a Job that contains tRunjob with this check box selected in Talend Administration Center.</td>
<td></td>
</tr>
<tr>
<td><strong>Warning:</strong> This option is incompatible with the Use or register a shared DB Connection option of database connection components. When tRunJob works together with a database connection component, enabling both options will cause your Job to fail.</td>
<td></td>
</tr>
</tbody>
</table>
### Die on child error
Clear this check box to execute the parent Job even though there is an error when executing the child Job.

### Transmit whole context
Select this check box to get all the context variables from the parent Job. Deselect it to get all the context variables from the child Job.

If this check box is selected when the parent and child Jobs have the same context variables defined:
- variable values for the parent Job will be used during the child Job execution if no relevant values are defined in the Context Param table.
- otherwise, values defined in the Context Param table will be used during the child Job execution.

### Context Param
You can change the value of selected context parameters. Click the [+ ] button to add the parameters defined in the Context tab of the child Job. For more information on context parameters, see Talend Studio User Guide.

The values defined here will be used during the child Job execution even if Transmit whole context is selected.

### Advanced settings

#### Propagate the child result to the output schema
Select this check box to propagate the output data stored in the buffer memory via the tBufferOutput component in the child Job to the output component in the parent Job.

This check box is cleared by default. It is invisible when the Use dynamic Job or Use an independent process to run subjob check box is selected.

#### Print Parameters
Select this check box to display the internal and external parameters in the Console.

#### tStatCatcher Statistics
Select this check box to gather the processing metadata at the Job level as well as at each component level.

### Global Variables

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

**CHILD_RETURN_CODE**: the return code of a child Job. This is an After variable and it returns an integer.

**CHILD_EXCEPTION_STACKTRACE**: the exception stack trace from a child Job. This is an After variable and it returns a string.
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component can be used as a standalone Job or can help clarifying complex Job by avoiding having too many sub-jobs all together in one Job.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connections</th>
<th>Outgoing links (from this component to another):</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row</strong>: Main.</td>
<td></td>
</tr>
<tr>
<td><strong>Trigger</strong>: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connections</th>
<th>Incoming links (from one component to this one):</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row</strong>: Main; Reject; Iterate.</td>
<td></td>
</tr>
<tr>
<td><strong>Trigger</strong>: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error; Synchronize; Parallelize.</td>
<td></td>
</tr>
</tbody>
</table>

For further information regarding connections, see *Talend Studio User Guide*.

### Calling a Job and passing the parameter needed to the called Job

This scenario describes a two-component Job named **ParentJob** that calls another Job named **ChildJob** to display the content of files specified in the **ParentJob** on the **Run** console.
Setting up the child Job

Procedure

1. Create a new Job ChildJob and add a **tFileInputDelimited** component and a **tLogRow** component to it.
2. Connect the **tFileInputDelimited** component to the **tLogRow** component using a **Row > Main** link.
3. Double-click the **tFileInputDelimited** component to open its **Basic settings** view.

4. Click in the **File Name** field and then press **F5** to open the **New Context Parameter** dialog box and configure the context variable.
5. In the Name field, enter a name for this new context variable, FilePath in this example.
6. In the Default value field, enter the full path to the default input file.
7. Click Finish to validate the context parameter setup and fill the File Name field with the context variable.

You can also create or edit a context parameter in the Context tab view beneath the design workspace. For more information, see Talend Studio User Guide.

8. Click the [...] button next to Edit schema to open the Schema dialog box where you can configure the schema manually.
9. In the dialog box, click the [+ ] button to add columns and name them according to the input file structure.

In this example, this component will actually read files defined in the parent Job, and these files contain up to five columns. Therefore, add five string type columns and name them Column1, Column2, Column3, Column4, and Column5 respectively, and then click OK to validate the schema configuration and close the Schema dialog box.

10. Double-click the tLogRow component and on its Basic settings view, select the Table option to view displayed content in table cells.

Setting up the parent Job

Procedure

1. Create a new Job ParentJob and add a tFileList component and a tRunJob component to it.
2. Connect the tFileList component to the tRunJob component using a Row > Iterate link.
3. Double-click the tFileList component to open its Basic settings view.
4. In the **Directory** field, specify the path to the directory that holds the files to be processed, or click the [...] button next to the field to browse to the directory.

   In this example, the directory is `D:/tRunJob_Input_Files` that holds three delimited files with up to five columns.

5. In the **FileList Type** list, select **Files**.

6. Select the **Use Glob Expressions as Filemask** check box, and then click the [+][+] button to add a line in the **Files** area and define a filter to match files. In this example, enter `"*.csv"` to retrieve all delimited files.

7. Double-click the **tRunJob** component to display its **Basic settings** view.

   8. Click the [...] button next to the **Job** field and in the pop-up dialog box, select the child Job you want to execute and click **OK** to close the dialog box. The name of the selected Job appears in the **Job** field.
9. In the **Context Param** area, click the [+] button to add a line and define the context parameter. The only context parameter defined in the child Job, named `FilePath`, appears in the **Parameters** cell.

10. Click in the **Values** cell, press **Ctrl+Space** on your keyboard to access the list of context variables, and select `tFileList_1.CURRENT_FILEPATH`.

    The corresponding context variable `((String)globalMap.get("tFileList_1_CURRENT_FILEPATH"))` appears in the **Values** cell.

    For more information on context variables, see *Talend Studio User Guide*.

**Executing the parent Job**

**Procedure**

1. Press **Ctrl+S** to save your Jobs.
2. Press **F6** to execute the parent Job.

```
<table>
<thead>
<tr>
<th>statistics</th>
<th>connecting to socket on port 33</th>
</tr>
</thead>
<tbody>
<tr>
<td>tLogRow_1</td>
<td></td>
</tr>
<tr>
<td>Column1</td>
<td>Column2</td>
</tr>
<tr>
<td>1</td>
<td>Jefferson</td>
</tr>
<tr>
<td>2</td>
<td>Washington</td>
</tr>
<tr>
<td>3</td>
<td>Eisenhower</td>
</tr>
<tr>
<td>4</td>
<td>Eisenhower</td>
</tr>
<tr>
<td>5</td>
<td>Arthur</td>
</tr>
</tbody>
</table>
```

The parent Job calls the child Job, which reads the files defined in the parent Job, and the content of the files is displayed on the Run console.
Running a list of child Jobs dynamically

This scenario describes a Job that calls two child Jobs dynamically. When called from the parent Job, each of these simple child Jobs displays a message on the console.

Setting up the child Jobs

Procedure

1. Create a new Job named ChildJob1, and add a tFixedFlowInput component and a tLogRow component to it.

![Diagram of tFixedFlowInput and tLogRow components connected]

2. Connect the tFixedFlowInput component to the tLogRow component using a Row > Main connection.

3. Double-click the tFixedFlowInput component to open its Basic settings view.

![Basic settings view of tFixedFlowInput component]

4. Click the [...] button next to Edit schema and in the pop-up dialog box, define the schema of the input data by adding one column Message of String type. When done, click OK to close the dialog box and click Yes when prompted to propagate the schema to the next component.

5. In the Mode area, select Use Inline Content(delimited file) and enter the message you want to show on the console in the Content field, Hello World! in this example.

6. Double-click the tLogRow component and on its Basic settings view, select the Table mode to display the execution result in table cells.

7. Create copy of this Job and name it ChildJob2, and enter another message in the Content field of the tFixedFlowInput component, Hello Talend! in this example.

Setting up the parent Job

Procedure

1. Create a new Job named ParentJob and add a tFixedFlowInput component, a tFlowToIterate component, and a tRunJob component to it.
2. Connect the **tFixedFlowInput** component to the **tFlowToIterate** component using a **Row > Main** connection and the **tFlowToIterate** component to the **tRunJob** component using a **Row > Iterate** connection.

3. Double-click the **tFixedFlowInput** component to open its **Basic settings** view.

4. Click the [...] button next to **Edit schema** and in the pop-up dialog box, define the schema of the input data by adding one column **JobName** of String type. When done, click **OK** to close the dialog box.

5. In the **Mode** area, select the **Use Inline Content(delimited file)** option and specify the names of the child Jobs to call from the parent Job in the **Content** field.

   - ChildJob1
   - ChildJob2

6. Double-click the **tRunJob** component to open its **Basic settings** view.
7. Select the Use dynamic job check box and in the Conext job field displayed, press Ctrl+Space and from the list of variables select the iterative global variable created by the tFlowTolterate component, tFlowTolterate_1.JobName in this example. The Context job field is then filled with ((String)globalMap.get("row1.JobName")). Upon each iteration, this variable will be resolved as the name of the Job to be called.

8. Click the [...] button next to the Job field and in the [Select Job] dialog box, select all the Jobs you want to run and click OK to close the dialog box. In this example, they are ChildJob1 and ChildJob2.

Executing the parent Job to run the child Jobs dynamically

Procedure

1. Save your child Jobs and parent Job.
2. Press F6 or click the Run button on the Run console to execute the Job.

As shown above, the child Jobs are called one after another and messages specified in the child Jobs are displayed on the console.
Propagating the buffered output data from the child Job to the parent Job

In this scenario, a three-component Job calls a two-component child Job and displays the buffered output data of the child Job, instead of the data from the input flow of the parent Job, on the console.

Setting up the child Job

About this task

Procedure

1. Create a Job named child, and add two components by typing their names on the design workspace or dropping them from the Palette to the design workspace:
   - a tFixedFlowInput, to generate a message
   - a tBufferOutput, to store the generated message in the buffer memory
2. Connect the tFixedFlowInput component to the tBufferOutput component using a Row > Main connection.
3. Double-click the tFixedFlowInput component to open its Basic settings view.
4. Click the [...] button next to Edit schema to open the [Schema] dialog box and define the schema of the input data. In this example, the schema has only one column message of the string type.
When done, click **OK** to validate the changes and then click **Yes** in the pop-up [Propagate] dialog box to propagate the schema to the next component.

5. In the **Mode** area, select **Use Single Table** option, and define the corresponding value for the `message` column in the **Values** table. In this example, the value is "message from the child job".

### Setting up the parent Job

**About this task**

**Procedure**

1. Create a Job, and add three components by typing their names on the design workspace or dropping them from the **Palette** to the design workspace:
   - a **tFixedFlowInput**, to generate a message
   - a **tRunJob**, to call the Job named `child`
   - a **tLogRow**, to display the execution result on the console

2. Connect the **tFixedFlowInput** component to the **tRunJob** component and the **tRunJob** component to the **tLogRow** component using the **Row > Main** connections.

3. Double-click the **tFixedFlowInput** component to open its **Basic settings** view.
4. Click the [...] button next to Edit schema to open the [Schema] dialog box and define the schema of the input data. In this example, the schema has only one column message of the string type.

When done, click OK to validate the changes.

5. In the Mode area, select the Use Single Table option, and define the corresponding value for the message column in the Values table. In this example, the value is "message from the parent job".

6. Click the tRunJob component and then click the Component tab to open its Basic settings view.

7. Click the Sync columns button and then click Yes in the pop-up [Propagate] dialog box to retrieve the schema from the preceding component.
8. Click the [...] button next to the Job field to open the [Repository Content] dialog box.

In the [Repository Content] dialog box, select the Job named child and then click OK to close the dialog box.

9. In the Advanced settings view of the tRunJob component, select the Propagate the child result to the output schema check box. With this check box selected, the buffered output of the child Job will be propagated to the output component.

**Executing the parent Job**

**Procedure**

1. Press Ctrl+S to save the Job.
2. Press F6 or click the Run button on the Run console to execute the Job.

The child Job is called and the message specified in the child Job, rather than the message defined in the parent Job, is displayed on the console.
tS3BucketCreate

Creates a bucket on Amazon S3.

“Bucket” is a term used by AWS for top level folders on S3, which can contain sub folders and store all your data (objects).

**tS3BucketCreate Standard properties**

These properties are used to configure tS3BucketCreate running in the Standard Job framework.

The Standard tS3BucketCreate component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Key</td>
<td>The Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit Getting Your AWS Access Keys.</td>
</tr>
<tr>
<td>Secret Key</td>
<td>The Secret Access Key, constituting the security credentials in combination with the access Key. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Inherit credentials from AWS role</td>
<td>Select this check box to obtain AWS security credentials from Amazon EC2 instance metadata. To use this option, the Amazon EC2 instance must be started and your Job must be running on Amazon EC2. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</td>
</tr>
<tr>
<td>Assume role</td>
<td>Select this check box and specify the values for the following parameters used to create a new assumed role session.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Role ARN</strong>: the Amazon Resource Name (ARN) of the role to assume.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Role session name</strong>: an identifier for the assumed role session.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Session duration (minutes)</strong>: the duration (in minutes) for which we want to have the assumed role session to be active.</td>
</tr>
<tr>
<td></td>
<td>For more information on assuming roles, see AssumeRole.</td>
</tr>
<tr>
<td>Region</td>
<td>Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (e.g. &quot;us-east-1&quot;) in the list. For more information about the AWS Region, see Regions and Endpoints.</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Bucket</td>
<td>Name of the bucket, namely the top level folder, to create.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| Config client | Select this check box to configure client parameters.  
**Client parameter**: select client parameters from the list.  
**Value**: enter the parameter value.  
Not available when Use existing connection is selected. |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| STS Endpoint  | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.  
This check box is available only when the Assume role check box is selected. |
| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is selected. |

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used alone or with other S3 components, e.g. tS3BucketExist.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  
Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. |
Related scenario

For **tS3BucketCreate** related scenarios, see Scenario: Verifying the absence of a bucket, creating it and listing all the S3 buckets on page 3324.
**tS3BucketDelete**

Deletes an empty bucket from Amazon S3.

“Bucket” is a term used by AWS for top level folders on S3, which can contain sub folders and store all your data (objects).

**tS3BucketDelete Standard properties**

These properties are used to configure tS3BucketDelete running in the Standard Job framework.

The Standard tS3BucketDelete component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use existing connection</strong></td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Access Key</strong></td>
<td>The Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit Getting Your AWS Access Keys.</td>
</tr>
<tr>
<td><strong>Secret Key</strong></td>
<td>The Secret Access Key, constituting the security credentials in combination with the access Key. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Inherit credentials from AWS role</strong></td>
<td>Select this check box to obtain AWS security credentials from Amazon EC2 instance metadata. To use this option, the Amazon EC2 instance must be started and your Job must be running on Amazon EC2. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</td>
</tr>
<tr>
<td><strong>Assume role</strong></td>
<td>Select this check box and specify the values for the following parameters used to create a new assumed role session.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Role ARN</strong>: the Amazon Resource Name (ARN) of the role to assume.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Role session name</strong>: an identifier for the assumed role session.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Session duration (minutes)</strong>: the duration (in minutes) for which we want to have the assumed role session to be active.</td>
</tr>
<tr>
<td></td>
<td>For more information on assuming roles, see AssumeRole.</td>
</tr>
<tr>
<td>Region</td>
<td>Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (e.g. &quot;us-east-1&quot;) in the list. For more information about the AWS Region, see Regions and Endpoints.</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Bucket</td>
<td>Name of the bucket, namely the top level folder, to delete.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| Config client | Select this check box to configure client parameters.  
Client parameter: select client parameters from the list. 
Value: enter the parameter value.  
Not available when Use existing connection is selected. |
|---------------|----------------------------------------------------------------------------------------------------------|
| STS Endpoint | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.  
This check box is available only when the Assume role check box is selected. |
| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is selected. |

**Usage**

| Usage rule | This component can be used alone or with other S3 components, e.g. tS3BucketList. |
| Dynamic settings | Click the [*] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  
Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. |
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenario

For **tS3BucketDelete** related scenarios, see Scenario: Verifying the absence of a bucket, creating it and listing all the S3 buckets on page 3324.
**tS3BucketExist**

Verifies if the specified bucket exists on Amazon S3.

"Bucket" is a term used by AWS for top level folders on S3, which can contain sub folders and store all your data (objects).

**tS3BucketExist Standard properties**

These properties are used to configure tS3BucketExist running in the Standard Job framework.

The Standard tS3BucketExist component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Key</strong></td>
<td>The Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit Getting Your AWS Access Keys.</td>
</tr>
<tr>
<td><strong>Secret Key</strong></td>
<td>The Secret Access Key, constituting the security credentials in combination with the access Key. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Inherit credentials from AWS role</strong></td>
<td>Select this check box to obtain AWS security credentials from Amazon EC2 instance metadata. To use this option, the Amazon EC2 instance must be started and your Job must be running on Amazon EC2. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</td>
</tr>
</tbody>
</table>
| **Assume role**         | Select this check box and specify the values for the following parameters used to create a new assumed role session.  
  - **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.  
  - **Role session name**: an identifier for the assumed role session.  
  - **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.  
  For more information on assuming roles, see AssumeRole. |
**Region**

Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (e.g. \"us-east-1\") in the list. For more information about the AWS Region, see [Regions and Endpoints](#).

**Bucket**

Name of the bucket, namely the top level folder, on the S3 server.

**Die on error**

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

---

**Advanced settings**

**Config client**

Select this check box to configure client parameters.

- **Client parameter**: select client parameters from the list.
- **Value**: enter the parameter value.

Not available when **Use existing connection** is selected.

**STS Endpoint**

Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.

This check box is available only when the **Assume role** check box is selected.

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

---

**Global Variables**

**BUCKET_EXIST**

The existence of a specified bucket. This is a Flow variable and it returns a boolean.

**BUCKET_NAME**

The name of a specified bucket. This is an After variable and it returns a string.

**ERROR_MESSAGE**

The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is selected.

---

**Usage**

**Usage rule**

This component is usually used with other S3 components, e.g. **tS3BucketCreate**.

**Dynamic settings**

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment.
where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Scenario: Verifying the absence of a bucket, creating it and listing all the S3 buckets

In this scenario, **tS3BucketExist** is used to verify the absence of a bucket, **tS3BucketCreate** to create that bucket upon confirmation, and **tS3BucketList** to list all the buckets on Amazon S3.

### Linking the components

**Procedure**

1. Drop **tS3Connection**, **tS3BucketExist**, **tS3BucketCreate**, **tS3BucketList**, **tIterateToFlow** and **tLogRow** onto the workspace.
2. Link **tS3Connection** to **tS3BucketExist** using the OnSubjobOk trigger.
3. Link **tS3BucketExist** to **tS3BucketCreate** using the Run if trigger.
4. Link **tS3BucketCreate** to **tS3BucketList** using the OnSubjobOk trigger.
5. Link **tS3BucketList** to **tIterateToFlow** using the Row > Iterate connection.
6. Link **tIterateToFlow** to **tLogRow** using the Row > Main connection.

![Diagram](image)

### Configuring the components

**Procedure**

1. Double-click **tS3Connection** to open its Basic settings view.
2. In the **Access Key** and **Secret Key** fields, enter the authentication credentials.

3. Double-click **tS3BucketExist** to open its **Basic settings** view.

4. Select the **Use existing connection** check box to reuse the connection.

5. In the **Bucket** field, enter the bucket name to check if it exists.

6. Double-click the **If** link to define the condition.

7. In the **Condition** box, enter the expression:

   ```java
   !((Boolean)globalMap.get("tS3BucketExist_1_BUCKET_EXIST"))
   ```

   This way, the rest of the Job will be executed if the specified bucket does not exist.

8. Double-click **tS3BucketCreate** to open its **Basic settings** view.

9. Double-click **tS3BucketList** to open its **Basic settings** view.
Select the **Use existing connection** check box to reuse the connection.

10. Double-click `tIterateToFlow` to open its **Basic settings** view.

11. Click **Edit schema** to open the schema editor.

12. In the **Mapping** area, press **Ctrl + Space** in the **Value** field to choose the variable `tS3BucketList_1_CURRENT_BUCKET_NAME`.

13. Double-click `tLogRow` to open its **Basic settings** view.

Select **Table (print values in cells of a table)** for a better display of the results.
Executing the Job

Procedure

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to run the Job.

   ```
   [statistics] connecting to socket on port 3661
   [statistics] connected
   -----------
   tLogRow_1
   =---------
   bucket_list
   =---------
   movies_andy
   | songs-andy
   -----------
   [statistics] disconnected
   ```

   As shown above, the bucket is created and all the buckets are listed.

3. Go to the S3 web console:

   ![S3 web console](image)

   As shown above, the bucket has been created on the S3 server.

4. Refresh the S3 Browser client:

   ![S3 Browser](image)

   This shows that the S3 Create action was performed successfully.
tS3BucketList

Lists all the buckets on Amazon S3.

"Bucket" is a term used by AWS for top level folders on S3, which can contain sub folders and store all your data (objects).

**tS3BucketList Standard properties**

These properties are used to configure tS3BucketList running in the Standard Job framework.

The Standard tS3BucketList component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Key</td>
<td>The Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit Getting Your AWS Access Keys.</td>
</tr>
<tr>
<td>Secret Key</td>
<td>The Secret Access Key, constituting the security credentials in combination with the access Key. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Inherit credentials from AWS role</td>
<td>Select this check box to obtain AWS security credentials from Amazon EC2 instance metadata. To use this option, the Amazon EC2 instance must be started and your Job must be running on Amazon EC2. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</td>
</tr>
</tbody>
</table>
| Assume role             | Select this check box and specify the values for the following parameters used to create a new assumed role session.  
  - **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.  
  - **Role session name**: an identifier for the assumed role session.  
  - **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.  
  For more information on assuming roles, see AssumeRole. |
Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (e.g. "us-east-1") in the list. For more information about the AWS Region, see Regions and Endpoints.

### Advanced settings

| Config client | Select this check box to configure client parameters.  
**Client parameter**: select client parameters from the list.  
**Value**: enter the parameter value.  
Not available when Use existing connection is selected. |
| STS Endpoint | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.  
This check box is available only when the Assume role check box is selected. |
| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

### Global Variables

| CURRENT_BUCKET_NAME | The current bucket name. This is a Flow variable and it returns a string. |
| NB_BUCKET | The number of buckets. This is an After variable and it returns an integer. |
| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

### Usage

| Usage rule | This component can be used alone or with other S3 components, e.g. tS3BucketDelete. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  
Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.  
For examples on using dynamic parameters, see Scenario: Reading data from databases through context- |
Related scenario

For **tS3BucketList** related scenarios, see **Scenario: Verifying the absence of a bucket, creating it and listing all the S3 buckets** on page 3324.
**tS3Close**

Shuts down a connection to Amazon S3, thus releasing the network resources.

**tS3Close Standard properties**

These properties are used to configure tS3Close running in the Standard Job framework.

The Standard tS3Close component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

| Component List | List of connection components. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

**Usage**

| Usage rule | As an end component, this component is to be used along with other S3 components, e.g. tS3Connection. |

| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |
Related scenario

For **tS3Close** related scenarios, see *Scenario: Listing files with the same prefix from a bucket* on page 3351.
**tS3Connection**

Establishes a connection to Amazon S3 to store and retrieve data.

**tS3Connection Standard properties**

These properties are used to configure tS3Connection running in the Standard Job framework.

The Standard tS3Connection component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Key</td>
<td>The Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit <a href="#">Getting Your AWS Access Keys</a>.</td>
</tr>
<tr>
<td>Secret Key</td>
<td>The Secret Access Key, constituting the security credentials in combination with the access key. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td>Inherit credentials from AWS role</td>
<td>Select this check box to obtain AWS security credentials from Amazon EC2 instance metadata. To use this option, the Amazon EC2 instance must be started and your Job must be running on Amazon EC2. For more information, see <a href="#">Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances</a>.</td>
</tr>
<tr>
<td>Assume role</td>
<td>Select this check box and specify the values for the following parameters used to create a new assumed role session.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Role ARN</strong>: the Amazon Resource Name (ARN) of the role to assume.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Role session name</strong>: an identifier for the assumed role session.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Session duration (minutes)</strong>: the duration (in minutes) for which we want to have the assumed role session to be active.</td>
</tr>
<tr>
<td></td>
<td>For more information on assuming roles, see AssumeRole.</td>
</tr>
<tr>
<td>Region</td>
<td>Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (e.g. “us-east-1”) in the list. For more information about the AWS Region, see <a href="#">Regions and Endpoints</a>.</td>
</tr>
<tr>
<td>Encrypt</td>
<td>Select this check box and from the <strong>Key type</strong> drop-down list displayed, select one of the following three options for encrypting the data on the client-side before sending</td>
</tr>
</tbody>
</table>
to Amazon S3. For more information, see Protecting Data Using Client-Side Encryption.

- **KMS-managed customer master key**: use a KMS-managed customer master key (CMK) for the client-side data encryption. In the **Key** field, you need to specify the AWS KMS customer master key ID (CMK ID).

- **Symmetric Master Key**: use a symmetric master key (256-bit AES secret key) for the client-side data encryption.
  - **Algorithm**: select the algorithm associated with the key from the list. By default, there is only one algorithm named **AES**.
  - **Encoding**: select the encoding type associated with the key from the list, either **Base64** or **X509**.
  - **Key** or **Key file**: specify the key or the path to the file that stores the key.

- **Asymmetric Master Key**: use an asymmetric master key (a 1024-bit RSA key pair) for the client-side data encryption.
  - **Algorithm**: select the algorithm associated with the key from the list. By default, there is only one algorithm named **RSA**.
  - **Public key file**: specify the path to the public key file.
  - **Private key file**: specify the path to the private key file.

### Advanced settings

<table>
<thead>
<tr>
<th>Use a custom region endpoint</th>
<th>Select this check box to use a custom endpoint and in the field displayed, specify the URL of the custom endpoint to be used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config client</td>
<td>Select this check box to configure client parameters. <strong>Client parameter</strong>: select client parameters from the list. <strong>Value</strong>: enter the parameter value.</td>
</tr>
<tr>
<td>STS Endpoint</td>
<td>Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from. This check box is available only when the <strong>Assume role</strong> check box is selected.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
### Global Variables

| **ERROR_MESSAGE** | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

### Usage

| Usage rule | As a start component, this component is to be used along with other S3 components. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

### Related scenario

For **tS3Connection** related scenarios, see Scenario: File exchanges with Amazon S3 on page 3358.
**tS3Copy**

Copies an Amazon S3 object from a source bucket to a destination bucket.

**tS3Copy Standard properties**

These properties are used to configure tS3Copy running in the Standard Job framework.

The Standard tS3Copy component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Key</td>
<td>Specify the Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit <a href="#">Getting Your AWS Access Keys</a>.</td>
</tr>
<tr>
<td>Secret Key</td>
<td>Specify the Secret Access Key, constituting the security credentials in combination with the access Key. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Inherit credentials from AWS role</td>
<td>Select this check box to obtain AWS security credentials from Amazon EC2 instance metadata. To use this option, the Amazon EC2 instance must be started and your Job must be running on Amazon EC2. For more information, see <a href="#">Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances</a>.</td>
</tr>
</tbody>
</table>
| Assume role               | Select this check box and specify the values for the following parameters used to create a new assumed role session.  
  - **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.  
  - **Role session name**: an identifier for the assumed role session.  
  - **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.  
  For more information on assuming roles, see [AssumeRole](#). |
<p>| Region                    | Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (e.g. &quot;us-east-1&quot;) in the list. For more information about the AWS Region, see <a href="#">Regions and Endpoints</a>. |</p>
<table>
<thead>
<tr>
<th><strong>Source Bucket</strong></th>
<th>Specify the name of the source bucket that contains the object to be copied.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source Key</strong></td>
<td>Specify the key of the object to be copied.</td>
</tr>
<tr>
<td><strong>Destination Bucket</strong></td>
<td>Specify the name of the destination bucket to which the object will be copied.</td>
</tr>
<tr>
<td><strong>Destination Key</strong></td>
<td>Specify the new key for the object after being copied to the destination bucket.</td>
</tr>
</tbody>
</table>

**Server-Side Encryption**
Select this check box to enable server-side encryption with Amazon S3-Managed Encryption Keys (SSE-S3) and use the 256-bit Advanced Encryption Standard (AES-256) cipher to encrypt your data.
For more information about the server-side encryption, see Protecting Data Using Server-Side Encryption.

**Use KMS**
Select this check box to enable server-side encryption with AWS KMS-Managed Keys (SSE-KMS) instead of Amazon S3-Managed Encryption Keys (SSE-S3).
This property is available when the Server-Side Encryption check box is selected.

**Customer Master Key**
Specify your own customer master key (CMK) that is created in the IAM console using AWS Key Management Service for SSE-KMS encryption. If not specified, the default CMK, which is created the first time you add an SSE-KMS encrypted object to the defined bucket in your region, will be used for SSE-KMS encryption.
This property is available when the Use KMS check box is selected.

**Die on error**
Select this check box to stop the execution of the Job when an error occurs.
Clear the check box to skip any rows on error and complete the process for error-free rows.

**Advanced settings**

| **Config client** | Select this check box and specify the client parameter(s) by clicking the [+] button to add as many rows as needed, each row for a client parameter, and then setting the value of the following fields for each parameter:  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STS Endpoint</strong></td>
<td>Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.</td>
</tr>
</tbody>
</table>
This check box is available only when the **Assume role** check box is selected.

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**ERROR_MESSAGE**

The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is selected.

### Usage

**Usage rule**

This component can be used as a standalone component.

### Scenario: Copying an S3 object from one bucket to another

This scenario describes a Job that uploads a new object to an existing empty S3 bucket `bucket-src`, then copies the object from the bucket `bucket-src` to another existing empty S3 bucket `bucket-dst`, finally lists the object in the bucket `bucket-dst` to see whether the object is successfully copied.
Setting up the Job

Procedure

1. Create a new Job and add a **tS3Connection** component, a **tS3Put** component, a **tS3Copy** component, a **tS3List** component, a **tIterateToFlow** component, and a **tLogRow** component by typing their names on the design workspace or dropping them from the Palette.

2. Link the **tS3List** component to the **tIterateToFlow** component using a **Row > Iterate** connection.

3. Link the **tIterateToFlow** component to the **tLogRow** component using a **Row > Main** connection.

4. Link the **tS3Connection** component to the **tS3Put** component using a **Trigger > On Subjob Ok** connection.

5. Do the same to link the **tS3Put** component to the **tS3Copy** component and the **tS3Copy** component to the **tS3List** component.

Configuring the components

Creating a connection to Amazon S3

Procedure

1. Double-click the **tS3Connection** component to open its **Basic settings** view on the **Component** tab.

2. In the **Access Key** and **Secret Key** fields, enter the authentication credentials required to access Amazon S3.

3. From the **Region** drop-down list, select an AWS region where the object will be uploaded and copied. In this example, we keep the default setting.

Uploading an object to an Amazon S3 bucket

Procedure

1. Double-click the **tS3Put** component to open its **Basic settings** view on the **Component** tab.
2. Select the **Use an existing connection** check box to reuse the Amazon S3 connection information you have defined in the **tS3Connection** component.

3. In the **Bucket** field, enter the name of the S3 bucket where the object will be uploaded. In this example, it is `bucket-src` that already exists in Amazon S3.

4. In the **Key** field, enter the key for the object to be uploaded. In this example, it is `tS3Copy_icon32_src.png`.

5. In the **File** field, browse to or enter the path to the object to be uploaded. In this example, it is `D:/tS3Copy_icon32.png`.

**Copying the uploaded object to another Amazon S3 bucket**

**Procedure**

1. Double-click the **tS3Copy** component to open its **Basic settings** view on the **Component** tab.

2. Select the **Use an existing connection** check box to reuse the Amazon S3 connection information you have defined in the **tS3Connection** component.

3. In the **Bucket** field in the **Source Configuration** area, enter the name of the bucket which contains the object to be copied. In this example, it is `bucket-src`.

4. In the **Key** field in the **Source Configuration** area, enter the key of the object to be copied. In this example, it is `tS3Copy_icon32_src.png`.

5. In the **Bucket** field in the **Destination Configuration** area, enter the name of the bucket to which the object will be copied. In this example, it is the empty one `bucket-dst` that already exists in Amazon S3.

6. In the **Key** field in the **Destination Configuration** area, enter the new key for the object after being copied to the destination bucket. In this example, it is `tS3Copy_icon32_dst.png`.

**Listing the object in the destination bucket**

**Procedure**

1. Double-click the **tS3List** component to open its **Basic settings** view on the **Component** tab.
2. Select the **Use an existing connection** check box to reuse the Amazon S3 connection information you have defined in the **tS3Connection** component.

3. Clear the **List all buckets objects** check box, and then click the [+ ] button to add one row in the **Bucket** table displayed and set the value for each column. In this example, *bucket-dist* for the **Bucket name** column and empty value for the **Key prefix** column, this way only the objects in the *bucket-dist* bucket will be listed.

4. Double-click the **tIterateToFlow** component to open its **Basic settings** view on the **Component** tab.

5. Click the [...] button next to **Edit schema** and in the pop-up schema dialog box define the schema by adding one column **ObjectList** of **String** type.

6. Click **OK** to save the changes and in the pop-up dialog box click **Yes** to accept the propagation.

7. Double-click the **tLogRow** component to open its **Basic settings** view on the **Component** tab.
8. In the **Mode** area, select **Table (print values in cells of a table)** for better readability of the result.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to run the Job.

```plaintext
[statistics] connecting to socket on port 3491
[statistics] connected
---------
| tLogRow_1 |
---------
| ObjectList |
---------
| tS3Copy_icon32_dst.png |
---------
[statistics] disconnected
```

As shown above, the uploaded object has been copied to the destination bucket successfully.
tS3Delete

Deletes a file from Amazon S3.

**tS3Delete Standard properties**

These properties are used to configure tS3Delete running in the Standard Job framework.

The Standard tS3Delete component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>Access Key</td>
<td>The Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit Getting Your AWS Access Keys.</td>
</tr>
<tr>
<td>Secret Key</td>
<td>The Secret Access Key, constituting the security credentials in combination with the access Key.</td>
</tr>
<tr>
<td></td>
<td>To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Inherit credentials from AWS role</td>
<td>Select this check box to obtain AWS security credentials from Amazon EC2 instance metadata. To use this option, the Amazon EC2 instance must be started and your Job must be running on Amazon EC2. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</td>
</tr>
<tr>
<td>Assume role</td>
<td>Select this check box and specify the values for the following parameters used to create a new assumed role session.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Role ARN</strong>: the Amazon Resource Name (ARN) of the role to assume.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Role session name</strong>: an identifier for the assumed role session.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Session duration (minutes)</strong>: the duration (in minutes) for which we want to have the assumed role session to be active.</td>
</tr>
<tr>
<td></td>
<td>For more information on assuming roles, see AssumeRole.</td>
</tr>
<tr>
<td>Region</td>
<td>Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (e.g. 'us-east-1') in the list. For more information about the AWS Region, see Regions and Endpoints.</td>
</tr>
</tbody>
</table>
Bucket | Name of the bucket, namely the top level folder, on Amazon S3.
---|---
Key | Name of the file to delete.
Die on error | This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

**Advanced settings**

| Config client | Select this check box to configure client parameters.  
|---|---  
| Client parameter: select client parameters from the list.  
| Value: enter the parameter value.  
| Not available when *Use existing connection* is selected. |
| STS Endpoint | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.  
| This check box is available only when the *Assume role* check box is selected. |
| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the *Die on error* check box is selected. |

**Usage**

| Usage rule | This component can be used alone or with other S3 components, e.g. *tS3BucketList*. |

| Dynamic settings | Click the [*] button to add a row in the table and fill the *Code* field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  
|---|---  
| Once a dynamic parameter is defined, the *Component List* box in the *Basic settings* view becomes unusable.  
| For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario:* |
Related scenario

For **tS3Delete** related scenarios, see *Scenario: Verifying the absence of a bucket, creating it and listing all the S3 buckets* on page 3324.
tS3Get

Retrieves a file from Amazon S3.

**tS3Get Standard properties**

These properties are used to configure tS3Get running in the Standard Job framework.

The Standard tS3Get component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Key</strong></td>
<td>The Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit Getting Your AWS Access Keys.</td>
</tr>
<tr>
<td><strong>Secret Key</strong></td>
<td>The Secret Access Key, constituting the security credentials in combination with the access Key. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Inherit credentials from AWS role</strong></td>
<td>Select this check box to obtain AWS security credentials from Amazon EC2 instance metadata. To use this option, the Amazon EC2 instance must be started and your Job must be running on Amazon EC2. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</td>
</tr>
<tr>
<td><strong>Assume role</strong></td>
<td>Select this check box and specify the values for the following parameters used to create a new assumed role session.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Role ARN</strong>: the Amazon Resource Name (ARN) of the role to assume.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Role session name</strong>: an identifier for the assumed role session.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Session duration (minutes)</strong>: the duration (in minutes) for which we want to have the assumed role session to be active. For more information on assuming roles, see AssumeRole.</td>
</tr>
<tr>
<td><strong>Region</strong></td>
<td>Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (e.g. &quot;us-east-1&quot;) in the list. For more information about the AWS Region, see Regions and Endpoints.</td>
</tr>
</tbody>
</table>
Encrypt

Select this check box and from the **Key type** drop-down list displayed, select one of the following three options for encrypting the data on the client-side before sending to Amazon S3. For more information, see [Protecting Data Using Client-Side Encryption](#).

- **KMS-managed customer master key**: use a KMS-managed customer master key (CMK) for the client-side data encryption. In the **Key** field, you need to specify the AWS KMS customer master key ID (CMK ID).

  - **Algorithm**: select the algorithm associated with the key from the list. By default, there is only one algorithm named **AES**.

  - **Encoding**: select the encoding type associated with the key from the list, either **Base64** or **X509**.

  - **Key or Key file**: specify the key or the path to the file that stores the key.

- **Symmetric Master Key**: use a symmetric master key (256-bit AES secret key) for the client-side data encryption.

  - **Algorithm**: select the algorithm associated with the key from the list. By default, there is only one algorithm named **AES**.

- **Asymmetric Master Key**: use an asymmetric master key (a 1024-bit RSA key pair) for the client-side data encryption.

  - **Algorithm**: select the algorithm associated with the key from the list. By default, there is only one algorithm named **RSA**.

  - **Public key file**: specify the path to the public key file.

  - **Private key file**: specify the path to the private key file.

<table>
<thead>
<tr>
<th>Bucket</th>
<th>Name of the bucket, namely the top level folder, on the S3 server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Name of the file saved on the S3 server.</td>
</tr>
<tr>
<td>File</td>
<td>Name of the file to be saved locally.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Config client</th>
<th>Select this check box to configure client parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client parameter</strong>: select client parameters from the list.</td>
<td></td>
</tr>
<tr>
<td><strong>Value</strong>: enter the parameter value.</td>
<td></td>
</tr>
<tr>
<td>Not available when <strong>Use existing connection</strong> is selected.</td>
<td></td>
</tr>
</tbody>
</table>
### STS Endpoint

Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.

This check box is available only when the **Assume role** check box is selected.

### tStatCatcher Statistics

Select this check box to collect log data at the component level.

---

### Global Variables

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is selected.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component can be used alone or with other S3 components, e.g. <strong>tS3Connection</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dynamic settings</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| Click the [+ ] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |

---

### Related scenario

For **tS3Get** related scenarios, see **Scenario: File exchanges with Amazon S3** on page 3358.
tS3List

Lists the files on Amazon S3 based on the bucket/file prefix settings.

**tS3List Standard properties**

These properties are used to configure tS3List running in the Standard Job framework.

The Standard tS3List component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Key</td>
<td>The Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit Getting Your AWS Access Keys.</td>
</tr>
<tr>
<td>Secret Key</td>
<td>The Secret Access Key, constituting the security credentials in combination with the access Key. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Inherit credentials from AWS role</td>
<td>Select this check box to obtain AWS security credentials from Amazon EC2 instance metadata. To use this option, the Amazon EC2 instance must be started and your Job must be running on Amazon EC2. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</td>
</tr>
</tbody>
</table>
| Assume role             | Select this check box and specify the values for the following parameters used to create a new assumed role session.  
  • **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.  
  • **Role session name**: an identifier for the assumed role session.  
  • **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.  
  For more information on assuming roles, see AssumeRole. |
| Region                  | Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (e.g. 'us-east-1') in the list. For more information about the AWS Region, see Regions and Endpoints. |
### List all bucket objects

Select this check box to list all the files on the S3 server.

**Key prefix:** enter the prefix of files to be listed. This way, only files with that prefix will be listed.

### Bucket

Click the [+] button to add one or more lines for defining the buckets and file prefixes.

**Bucket name:** name of the bucket, namely the top level folder, whose files will be listed.

**Key prefix:** prefix of files to be listed.

Not available when **List all bucket objects** is selected.

### Die on error

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

### Advanced settings

#### Config client

Select this check box to configure client parameters.

**Client parameter:** select client parameters from the list.

**Value:** enter the parameter value.

Not available when **Use existing connection** is selected.

#### STS Endpoint

Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.

This check box is available only when the **Assume role** check box is selected.

#### tStatCatcher Statistics

Select this check box to collect log data at the component level.

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CURRENT_BUCKET_NAME</strong></td>
<td>The current bucket name. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td><strong>CURRENT_KEY</strong></td>
<td>The current key. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td><strong>NB_BUCKET</strong></td>
<td>The number of buckets. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_BUCKET_OBJECT</strong></td>
<td>The number of objects in all buckets. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is selected.</td>
</tr>
</tbody>
</table>
## Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used alone or with other S3 components, e.g. <strong>tS3Delete</strong>.</th>
</tr>
</thead>
</table>

### Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

## Scenario: Listing files with the same prefix from a bucket

In this scenario, **tS3List** is used to list all the files in a bucket which have the same prefix.

There are such files in this bucket:

![S3 Browser](image)

For how to create a bucket and put files into it, see Scenario: Verifying the absence of a bucket, creating it and listing all the S3 buckets on page 3324 and Scenario: File exchanges with Amazon S3 on page 3358.

## Linking the components

### Procedure

1. Drop **tS3Connection**, **tS3List**, **tIterateToFlow**, **tLogRow** and **tS3Close** onto the workspace.
2. Link **tS3Connection** to **tS3List** using the **OnSubjobOk** trigger.
3. Link `tS3List` to `tIterateToFlow` using the `Row > Iterate` connection.
4. Link `tIterateToFlow` to `tLogRow` using the `Row > Main` connection.
5. Link `tS3List` to `tS3Close` using the `OnSubjobOk` trigger.

**Configuring the components**

**Procedure**

1. Double-click `tS3Connection` to open its **Basic settings** view.

2. In the **Access Key** and **Secret Key** fields, enter the authentication credentials.

3. Double-click `tS3List` to open its **Basic settings** view.
4. Select the **Use existing connection** check box to reuse the connection.

5. In the **Bucket** area, click the [*] button to add one line.

6. In the **Bucket name** and **Key prefix** fields, enter the bucket name and file prefix. This way, only files with the specified prefix will be listed.

7. Double-click **tIterateToFlow** to open its **Basic settings** view.

8. Click **Edit schema** to open the schema editor.

   Click the [*] button to add one column, namely **file_list** of the String type.
   Click **Ok** to validate the setup and close the schema editor.

9. In the **Mapping** area, press **Ctrl + Space** in the **Value** field to choose the variable `tS3List_1_CURR\_ENT_KEY`.

10. Double-click **tLogRow** to open its **Basic settings** view.
Select **Table (print values in cells of a table)** for a better display of the results.

11. Double-click **tS3Close** to open its **Basic settings** view.

There is no need to select a connection component as the only one is selected by default.

**Executing the Job**

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to run the Job.

   ![Statistics](image)

   As shown above, only the files with the prefix "in" are listed.
tS3Put

Uploads data onto Amazon S3 from a local file or from cache memory via the streaming mode.

**tS3Put Standard properties**

These properties are used to configure tS3Put running in the Standard Job framework.

The Standard tS3Put component belongs to the Cloud family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Use existing connection</th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Key</td>
<td>The Access Key ID that uniquely identifies an AWS Account. For how to get your Access Key and Access Secret, visit <a href="#">Getting Your AWS Access Keys</a>.</td>
</tr>
<tr>
<td>Secret Key</td>
<td>The Secret Access Key, constituting the security credentials in combination with the access Key. To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td>Inherit credentials from AWS role</td>
<td>Select this check box to obtain AWS security credentials from Amazon EC2 instance metadata. To use this option, the Amazon EC2 instance must be started and your Job must be running on Amazon EC2. For more information, see <a href="#">Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances</a>.</td>
</tr>
<tr>
<td>Assume role</td>
<td>Select this check box and specify the values for the following parameters used to create a new assumed role session.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Role ARN</strong>: the Amazon Resource Name (ARN) of the role to assume.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Role session name</strong>: an identifier for the assumed role session.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Session duration (minutes)</strong>: the duration (in minutes) for which we want to have the assumed role session to be active.</td>
</tr>
<tr>
<td></td>
<td>For more information on assuming roles, see <a href="#">AssumeRole</a>.</td>
</tr>
<tr>
<td>Region</td>
<td>Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (e.g. &quot;us-east-1&quot;) in the list. For more information about the AWS Region, see <a href="#">Regions and Endpoints</a>.</td>
</tr>
</tbody>
</table>
Encrypt

Select this check box and from the **Key type** drop-down list displayed, select one of the following three options for encrypting the data on the client-side before sending to Amazon S3. For more information, see [Protecting Data Using Client-Side Encryption](#).

- **KMS-managed customer master key**: use a KMS-managed customer master key (CMK) for the client-side data encryption. In the **Key** field, you need to specify the AWS KMS customer master key ID (CMK ID).
  - **Algorithm**: select the algorithm associated with the key from the list. By default, there is only one algorithm named **AES**.
  - **Encoding**: select the encoding type associated with the key from the list, either **Base64** or **X509**.
  - **Key or Key file**: specify the key or the path to the file that stores the key.

- **Symmetric Master Key**: use a symmetric master key (256-bit AES secret key) for the client-side data encryption.
  - **Algorithm**: select the algorithm associated with the key from the list. By default, there is only one algorithm named **AES**.
  - **Encoding**: select the encoding type associated with the key from the list, either **Base64** or **X509**.
  - **Key or Key file**: specify the key or the path to the file that stores the key.

- **Asymmetric Master Key**: use an asymmetric master key (a 1024-bit RSA key pair) for the client-side data encryption.
  - **Algorithm**: select the algorithm associated with the key from the list. By default, there is only one algorithm named **RSA**.
  - **Public key file**: specify the path to the public key file.
  - **Private key file**: specify the path to the private key file.

---

<table>
<thead>
<tr>
<th>Bucket</th>
<th>Name of the bucket, namely the top level folder, on the S3 server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>The name of the file that will hold uploaded data on the S3 server.</td>
</tr>
</tbody>
</table>
| File or Stream | The path to the local file to be uploaded or the input stream from which data will be uploaded.  
For an example of how to use stream, see [Scenario 2: Reading data from a remote file in streaming mode](#) on page 1015. |
| Server-Side Encryption | Select this check box to enable server-side encryption with Amazon S3-Managed Encryption Keys (SSE-S3) and use the 256-bit Advanced Encryption Standard (AES-256) cipher to encrypt your data.  
For more information about the server-side encryption, see [Protecting Data Using Server-Side Encryption](#). |
| Use KMS | Select this check box to enable server-side encryption with AWS KMS-Managed Keys (SSE-KMS) instead of Amazon S3-Managed Encryption Keys (SSE-S3). |
This property is available when the **Server-Side Encryption** check box is selected.

### Customer Master Key

Specify your own customer master key (CMK) that is created in the IAM console using AWS Key Management Service for SSE-KMS encryption. If not specified, the default CMK, which is created the first time you add an SSE-KMS encrypted object to the defined bucket in your region, will be used for SSE-KMS encryption.

This property is available when the **Use KMS** check box is selected.

### Die on error

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

### Advanced settings

| Config client | Select this check box to configure client parameters.  
**Client parameter:** select client parameters from the list.  
**Value:** enter the parameter value.  
Not available when **Use existing connection** is selected. |
|---|---|
| STS Endpoint | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.  
This check box is available only when the **Assume role** check box is selected. |
| Part size (Mb) | Specify the part size in MB when uploading a large file in parts. For more information about S3 multipart upload, see [Multipart Upload Overview](#). |
| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

### Global Variables

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is selected. |

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used alone or with other S3 components, e.g. <strong>tS3Connection</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the <strong>[+]</strong> button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having...</td>
</tr>
</tbody>
</table>
the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Scenario: File exchanges with Amazon S3

In this scenario, **tS3Put** is used to send a file to the Amazon S3 server while **tS3Get** is used to get that file.

#### Linking the components

**Procedure**

1. Drop **tS3Connection**, **tS3Put** and **tS3Get** onto the workspace.
2. Link **tS3Connection** to **tS3Put** using the **OnSubjobOk** trigger.
3. Link **tS3Put** to **tS3Get** using the **OnSubjobOk** trigger.

#### Configuring the components

**Procedure**

1. Double-click **tS3Connection** to open its Basic settings view.
2. In the **Access Key** and **Secret Key** fields, enter the authentication credentials.
3. Double-click **tS3Put** to open its **Basic settings** view.

4. Select the **Use existing connection** check box to reuse the connection.
5. In the **Bucket** field, enter the bucket name.
6. In the **Key** field, enter the name of the file to be saved on the S3 server.
7. In the **File** field, enter the file to be uploaded to the S3 server.

8. Double-click **tS3Get** to open its **Basic settings** view.

9. Select the **Use existing connection** check box to reuse the connection.
10. In the **Bucket** field, enter the bucket name.
11. In the **Key** field, enter the name of the file to be retrieved from the S3 server.
12. In the **File** field, enter the local name of the retrieved file.

**Executing the Job**

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to run the Job.
3. Go to the S3 web console:

As shown above, the file has been put on to the S3 server.

4. Refresh the S3 Browser client:

This shows that the S3 Put action was performed successfully.

5. Go to the folder where the retrieved file is saved locally:

As shown above, the remote file is retrieved to the local disk, proof that the S3 Get action was performed successfully.
tSageX3Input

Extracts data from a given Sage X3 system.

This component leverages the Web service provided by a given Sage X3 Web server to extract data from the Sage X3 system (the X3 server).

**tSageX3Input Standard properties**

These properties are used to configure tSageX3Input running in the Standard Job framework.

The Standard tSageX3Input component belongs to the Business family.

The component in this framework is generally available.

**Basic settings**

| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click Edit schema to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td><strong>No property data stored centrally.</strong></td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td><strong>Select the repository file where properties are stored. The fields that come after are pre-filled in using the fetched data.</strong></td>
</tr>
<tr>
<td><strong>Endpoint address</strong></td>
<td><strong>Type in the address of the Web service provided by the given Sage X3 Web server.</strong></td>
</tr>
</tbody>
</table>
| **Username and Password**| **Type in the Web service user authentication data that you have defined for configuring the Sage X3 Web server.**

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
<table>
<thead>
<tr>
<th><strong>Language</strong></th>
<th>Type in the name of the X3 language code used to start a connection group.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pool alias</strong></td>
<td>Type in the name of the connection pool that distributes the received requests to available connections. This name was given from the Sage X3 configuration console.</td>
</tr>
</tbody>
</table>
| **Request config** | Type in the configuration string if you want to retrieve the debug or trace information. For example, the string could be:  
RequestConfigDebug="adxwss.trace.on=on";  
If you need to use several strings, separate them with a & for example,  
RequestConfigDebug="adxwss.trace.on=on&adxwss.trace.size=16384";  
**Warning:**  
A third party tool is needed to retrieve this kind of information. |
| **Publication name** | Type in the publication name of the published object, list or sub-program you want your Studio to access. |
| **Action** | Select in the list the action to be carried on.  
• **getAllDetails**: returns the details of all the records.  
• **getSummary**: returns the summary of each record.  
• **getDetail**: returns the details of the records that meet the conditions configured in the **Query condition** table. |
| **Mapping** | Complete this table to map the variable elements of the object, the sub-program or the list set in the given Sage X3 Web server. The columns to be completed include:  
**Column**: the columns defined in the schema editor for this component.  
**Group ID**: the identifier of each variable element group. For example, a variable element group could represent one of attributes of an object.  
**Field name**: the field name of each variable element. |
| **Conditions** | Complete this table to set up the query condition(s). The columns to be completed include:  
**Key**: the names of the variable elements used as the key for data extraction.  
**Value**: the value of the given key field used to extract the corresponding data.  
Available when **getDetail** is selected in the **Action** list. |
| **Query condition** | Select this check box to set up the query condition(s). The columns to be completed include:  
**Key**: the names of the variable elements used as the key for data extraction. |
Value: the value of the given key field used to extract the corresponding data.
Not available when getDetail is selected in the Action list.

Limit
Type in a number to indicate the maximum row count of the data to be extracted.
Not available when getDetail is selected in the Action list.

Advanced settings

Global Variables

Global Variables
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.
For further information about variables, see Talend Studio User Guide.

Usage

Usage rule
Usually used as a Start component. An output component is required.

Scenario: Using query key to extract data from a given Sage X3 system

This scenario describes a two-component Job used to extract one row of data from a given Sage X3 system. The object method is to be called, that means the variable elements of this object thus are attributes. The data used in this scenario can be found in the example provided by Sage X3.
Setting up the Job

Procedure

1. Drop the tSageX3Input component and the tLogRow components onto the workspace from Palette.
2. Connect the tSageX3Input component to the tLogRow component using a Row > Main link.

Configuring the schema of tSageX3Input

Procedure

1. Double-click tSageX3Input to set its properties in the Basic Settings view.

   ![tSageX3Input_1]

   In the Schema field, select Built-In and click the three-dot button next to Edit schema to open the schema editor.

2. In the Schema field, select Built-In and click the three-dot button next to Edit schema to open the schema editor.
3. In this editor, click the plus button 12 times beneath the schema table to add 12 rows into this table.

4. Type in the names you want to use for each row. In this example, these rows are named after the publication names of the object attributes set in the Sage X3 Web server. These columns are used to map the corresponding attribute fields in the Sage X3 system.

5. In the Type column, click the IMG row to display its drop-down list.

6. From the drop-down list, select List as this attribute appears twice or even more and do the same to switch the types of the TIT2NBLIG row, the ITMLNK row and the ZITMLNK row to List as well for the same reason.

7. Click OK to validate this change and accept the propagation prompted by a pop-up dialog box.

**Configuring the connection to the Sage X3 Web server**

**Procedure**

1. In the Endpoint address field, type in the URL address of the Web service provided by the Sage X3 Web server. In this example, it is http://10.42.20.168:28880/adxwsvc/services/CAdxWebServiceXmlCC

2. In the User field, type in the user name of the given Sage X3. In this example, it is ERP.

3. In the Language field, type in the name of the X3 language code used to start a connection group. In this example, it is FRA.

4. In the Pool alias field, type in the name of connection pool to be used. In this example, this connection pool is called TALEND.

5. In the Publication name field, type in the publication name of the object to be called. In this scenario, the publication name is ITMDET.

**Setting up the mapping and configuring the query condition**

**Procedure**

1. In the Group ID column and the Field name column of the Mapping table, type in values corresponding to the attribute group IDs and the attribute publication names defined in the Sage X3 Web server. In this example, the values are presented in the figure below.
1. In the **Mapping** table, the **Column** column has been filled automatically with the columns you created in the schema editor.

2. Select the **Query condition** check box to activate the **Conditions** table.

3. Under the **Conditions** table, click the plus button to add one row into the table.

4. In the **Key** column, type in the publication name associated with the object attribute you need to extract data from.

5. In the **Value** column, type in the value of the attribute you have selected as the key of the data extraction. In this scenario, it is **CONT000059**, one of the product references.

### Job execution

**Procedure**

1. Press **Ctrl+S** to save your Job.

2. Press **F6** or click **Run** on the **Run** tab to execute the Job.

   The results are displayed on the **Run** console:
tSageX3Output

Writes data into a given Sage X3 system.

This component connects to the Web service provided by a given Sage X3 Web server and therefrom insert, update or delete data in the Sage X3 system (the X3 server).

tSageX3Output Standard properties

These properties are used to configure tSageX3Output running in the Standard Job framework.

The Standard tSageX3Output component belongs to the Business family.

The component in this framework is generally available.

Basic settings

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:
- View schema: choose this option to view the schema only.
- Change to built-in property: choose this option to change the schema to Built-in for local changes.
- Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the Repository Content window.
Click Sync columns to retrieve the schema from the previous component connected in the Job. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file where properties are stored. The fields that come after are pre-filled in using the fetched data.</td>
<td></td>
</tr>
<tr>
<td>Endpoint address</td>
<td>Type in the address of the Web service provided by the given Sage X3 Web server.</td>
</tr>
</tbody>
</table>
| Username and Password | Type in the Web service user authentication data that you have defined for configuring the Sage X3 Web server.
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter |
<table>
<thead>
<tr>
<th><strong>Language</strong></th>
<th>Type in the name of the X3 language code used to start a connection group.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pool alias</strong></td>
<td>Type in the name of the connection pool that distributes the received requests to available connections. This name was given from the Sage X3 configuration console.</td>
</tr>
</tbody>
</table>
| **Request config** | Type in the configuration string if you want to retrieve the debug or trace information.  
For example, the string could be: "RequestConfigDebug=adxwss.trace.on=on";  
If you need use several strings, separate them with a ", for example,  
RequestConfigDebug=adxwss.trace.on=on&adxwss.trace.size=16384";  
**Warning:**  
A third party tool is needed to retrieve this kind of information. |
| **Publication name** | Type in the publication name of the published object, list or sub-program you want your Studio to access. |
| **Action** | You can do any of the following operations on the data in a Sage X3 system:  
**Insert:** insert data  
**Update:** update data  
**Delete:** delete data |
| **Mapping** | Complete this table to map the variable elements of the object, the list or the sub-program your Studio access. Only the elements you need to conduct the data action of your interest on are selected and typed in for the purpose of mapping. The columns to be completed include:  
**Column:** the columns defined in the schema editor for this component.  
**Key:** the variable element used as key for data insertion, update or deletion. Select the corresponding check box if a variable element is the key.  
**Group ID:** the identifier of each variable element group. For example, a variable element group could represent one of attributes of an object.  
**Field name:** the field name of each selected variable element. |
| **Advanced settings** |  
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |
Global Variables

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

Usage

| Usage rule | Usually used as an output component. An input component is required. |

Scenario: Using a Sage X3 Web service to insert data into a given Sage X3 system

This scenario describes a two-component Job used to generate one row of data and insert the data into a given Sage X3 system. You can find the data used in this scenario in the example provided by Sage X3. The Sage X3 Web service is used to access an object.

Setting up the Job

**Procedure**

1. Drop the **tFixedFlowInput** and the **tSageX3Output** components onto the workspace from **Palette**.
2. Connect the **tFixedFlowInput** component to the **tSageX3Output** component using a **Row > Main** connection.

Configuring the schema for the input data

**Procedure**

1. Double-click the **tFixedFlowInput** component to set its Basic Settings in the Component view.
2. Click the three-dot button next to Edit schema to open the schema editor.

3. In the schema editor and then under the schema table, click the plus button four times to add four rows.

4. Click OK to validate this changes and then accept the propagation prompted by the pop-up dialog box. The four rows appear automatically in the Values table of the Component view.

5. In the Values table within the Mode area, type in the values for each of the four rows in the Value column. In this scenario, the values downward are: CONTS00059, Screen 24" standard 16/10, Screen 24" standard 28/10, 2.

**Warning:**

*These values in the Value column must be put between quotation marks.*

**Setting up the connection to the Sage X3 Web server**

**Procedure**

1. Double-click tSageX3Output to set its properties from the Basic Settings view.
2. In the **Endpoint address** field, type in the URL address of the Web service provided by the Sage X3 Web server. In this example, it is `http://10.42.20.168:28880/adxwsvc/services/CAdxWebServiceXmlCC`.

3. In the **User** field, type in the user name of the given Sage X3. In this example, it is `ERP`.

4. In the **Language** field, type in the name of the X3 language code used to start a connection group. In this example, it is `FRA`.

5. In the **Pool alias** field, type in the name of connection pool to be used. In this example, this connection pool is called `TALEND`.

6. In the **Publication name** field, type in the publication name of the object to be called. In this scenario, the publication name is `ITMDET`.

7. In the **Action** field, select **insert** from the drop-down list.

### Setting up the mapping

#### Procedure

1. In the **Field name** column of the **Mapping** table, type in the field names of the attributes the selected data action is exercised on.

2. In the **Group ID** column of the **Mapping** table, type in values corresponding to group IDs of the selected attributes. These IDs are defined in the Sage X3 Web server.

### Note:
In the **Mapping** table, the **Column** column has been filled automatically with the columns retrieved from the schema of the preceding component.

**Job execution**

Press CTRL+S to save your Job and press **F6** to execute it.

To verify the data that you inserted in this scenario, you can use the **tSageX3Input** component to read the concerned data from the Sage X3 server.

For further information about how to use the **tSageX3Input** component to read data, see Scenario: Using query key to extract data from a given Sage X3 system on page 3363.
tSalesforceBulkExec

Bulk-loads data in a given file into a Salesforce object.

The tSalesforceOutputBulk and tSalesforceBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used to feed the Salesforce database. These two steps are fused together in the tSalesforceOutputBulkExec component. The advantage of using two separate steps is that the data can be transformed before it is loaded into Salesforce.

tSalesforceBulkExec Standard properties

These properties are used to configure tSalesforceBulkExec running in the Standard Job framework. The Standard tSalesforceBulkExec component belongs to the Business and the Cloud families. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
</tbody>
</table>

| **Connection Component** | Select the component that opens the database connection to be reused by this component. |

| **Connection type** | Select the type of the connection from the drop-down list. The connection properties will vary according to the connection type you choose. |
|                    | • **Basic**: select this option to access Salesforce by entering your Salesforce username and password. |
|                    | • **OAuth**: select this option to access Salesforce using OAuth (Open Authorization) 2.0 protocol to authenticate the user. From the Oauth2 flow type drop-down list displayed, you need to choose an OAuth 2.0 authentication flow type, either Json Web Token Flow or Implicit Flow (Deprecated) (User Agent Flow). This way, you need to create a connected app in Salesforce to configure a Salesforce authentication provider. |
For more information about the OAuth authentication flow, see Authenticate Apps with OAuth.

For more information about how to create a connected app in Salesforce, see Create a Connected App.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Available for</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Id</td>
<td>The Salesforce username. This property is available only when the <strong>Basic</strong> connection type is selected.</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td>The Salesforce password associated with the username. This property is available only when the <strong>Basic</strong> connection type is selected.</td>
<td></td>
</tr>
<tr>
<td>Security Token</td>
<td>The Salesforce security token. For more information, see Reset Your Security Token. This property is available only when the <strong>Basic</strong> connection type is selected.</td>
<td></td>
</tr>
<tr>
<td>Issuer</td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see Create a Connected App. This property is available only when the <strong>OAuth Json Web Token Flow</strong> type is selected.</td>
<td></td>
</tr>
<tr>
<td>Subject</td>
<td>The Salesforce username. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
<td></td>
</tr>
<tr>
<td>Expiration time (in seconds)</td>
<td>The expiration time of the assertion (in seconds) within five minutes. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
<td></td>
</tr>
<tr>
<td>Key store</td>
<td>The path to the keystore file in Java Keystore (JKS) format. The keystore file can be generated by creating a certificate signed by Salesforce and then exporting it to keystore. For more information, see Generate a Self-Signed Certificate. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
<td></td>
</tr>
<tr>
<td>Key store password</td>
<td>The keystore password. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
<td></td>
</tr>
<tr>
<td>Certificate alias</td>
<td>The unique name of the certificate signed by Salesforce. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
<td></td>
</tr>
<tr>
<td><strong>Client Id</strong></td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see <a href="#">Create a Connected App</a>. This property is available only when the OAuth Implicit Flow type is selected.</td>
<td></td>
</tr>
<tr>
<td><strong>Client Secret</strong></td>
<td>The OAuth Consumer Secret, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see <a href="#">Create a Connected App</a>. This property is available only when the OAuth Implicit Flow type is selected.</td>
<td></td>
</tr>
<tr>
<td><strong>Callback Host</strong></td>
<td>The host value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the <a href="#">API (Enable OAuth Settings)</a> area of the connected app detail page in Salesforce. This property is available only when the OAuth Implicit Flow type is selected.</td>
<td></td>
</tr>
<tr>
<td><strong>Callback Port</strong></td>
<td>The port value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the <a href="#">API (Enable OAuth Settings)</a> area of the connected app detail page in Salesforce. This property is available only when the OAuth Implicit Flow type is selected.</td>
<td></td>
</tr>
<tr>
<td><strong>Token File</strong></td>
<td>The path to the token file that stores the refresh token used to get the access token without authorization. This property is available only when the OAuth Implicit Flow type is selected.</td>
<td></td>
</tr>
<tr>
<td><strong>Module Name</strong></td>
<td>Click the [...] button next to the field and in the dialog box displayed, select the module that will be used or select the <strong>Use custom object</strong> check box and specify the module name in the <strong>Object Name</strong> field.</td>
<td></td>
</tr>
</tbody>
</table>
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No.
upon completion and choose this schema metadata again in the [Repository Content] window.

Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

<table>
<thead>
<tr>
<th>Output Action</th>
<th>Select one of the following operations to be performed from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>INSERT</strong>: insert one or more new records into Salesforce.</td>
</tr>
<tr>
<td></td>
<td>• <strong>UPDATE</strong>: update one or more existing records in Salesforce.</td>
</tr>
<tr>
<td></td>
<td>• <strong>UPSERT</strong>: create new records and update existing records. In the <strong>Upsert Key Column</strong> field displayed, specify the key column for the upsert operation.</td>
</tr>
<tr>
<td></td>
<td>• <strong>DELETE</strong>: delete one or more records in Salesforce.</td>
</tr>
</tbody>
</table>

| Bulk File Path | Specify the path to the file that stores the data to be processed. |

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Salesforce URL</strong></th>
<th>The Webservice URL required to connect to Salesforce.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API version</strong></td>
<td>The Salesforce API version. This property is available only when the OAuth connection type is selected.</td>
</tr>
<tr>
<td><strong>Need compression</strong></td>
<td>Select this check box to activate SOAP message compression, which can result in increased performance levels.</td>
</tr>
<tr>
<td><strong>Trace HTTP message</strong></td>
<td>Select this check box to output the HTTP interactions on the console.</td>
</tr>
<tr>
<td><strong>Client Id</strong></td>
<td>Enter the ID of the real user to differentiate between those who use the same account and password to access the Salesforce website.</td>
</tr>
<tr>
<td><strong>Timeout</strong></td>
<td>Enter the intended number of query timeout in milliseconds in Salesforce.</td>
</tr>
<tr>
<td><strong>Use Proxy</strong></td>
<td>Select this check box to use a proxy server, and in the <strong>Host</strong>, <strong>Port</strong>, <strong>User Id</strong>, and <strong>Password</strong> fields displayed, specify the connection parameters of the proxy server.</td>
</tr>
<tr>
<td><strong>Concurrency Mode</strong></td>
<td>Select the concurrency mode for the job.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Parallel</strong>: process batches in parallel mode.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Serial</strong>: process batches in serial mode.</td>
</tr>
<tr>
<td><strong>Rows to Commit</strong></td>
<td>Specify the number of lines per data batch to be processed.</td>
</tr>
</tbody>
</table>
### tSalesforceBulkExec

<table>
<thead>
<tr>
<th><strong>Bytes to Commit</strong></th>
<th>Specify the number of bytes per data batch to be processed.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Wait Time Check Batch State</strong></td>
<td>Specify the wait time (in milliseconds) for checking whether the batches in a Job have been processed until all batches are finally processed.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th><strong>NB_LINE</strong></th>
<th>The number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_SUCCESS</strong></td>
<td>The number of rows successfully processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_REJECT</strong></td>
<td>The number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

### Usage

| **Usage rule** | This component is more commonly used with the tSalesforceOutputBulk component. Used together, they gain performance while feeding or modifying information in Salesforce. |

### Related scenario

No scenario is available for this component yet.
**tSalesforceConnection**

Opens a connection to Salesforce.

**tSalesforceConnection Standard properties**

These properties are used to configure tSalesforceConnection running in the Standard Job framework.

The Standard tSalesforceConnection component belongs to the Business and the Cloud families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td><strong>Connection type</strong></td>
<td>Select the type of the connection from the drop-down list. The connection properties will vary according to the connection type you choose.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Basic:</strong> select this option to access Salesforce by entering your Salesforce username and password.</td>
</tr>
<tr>
<td></td>
<td>• <strong>OAuth:</strong> select this option to access Salesforce using OAuth (Open Authorization) 2.0 protocol to authenticate the user. From the OAuth2 flow type drop-down list displayed, you need to choose an OAuth 2.0 authentication flow type, either Json Web Token Flow or Implicit Flow (Deprecated) (User Agent Flow). This way, you need to create a connected app in Salesforce to configure a Salesforce authentication provider.</td>
</tr>
<tr>
<td></td>
<td>For more information about the OAuth authentication flow, see Authenticate Apps with OAuth.</td>
</tr>
<tr>
<td></td>
<td>For more information about how to create a connected app in Salesforce, see Create a Connected App.</td>
</tr>
<tr>
<td><strong>User Id</strong></td>
<td>The Salesforce username.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the <strong>Basic</strong> connection type is selected.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>The Salesforce password associated with the username.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Security Token</td>
<td>The Salesforce security token. For more information, see <a href="#">Reset Your Security Token</a>. This property is available only when the Basic connection type is selected.</td>
</tr>
<tr>
<td>Issuer</td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see <a href="#">Create a Connected App</a>. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Subject</td>
<td>The Salesforce username. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Expiration time (in seconds)</td>
<td>The expiration time of the assertion (in seconds) within five minutes. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Key store</td>
<td>The path to the keystore file in Java Keystore (JKS) format. The keystore file can be generated by creating a certificate signed by Salesforce and then exporting it to keystore. For more information, see <a href="#">Generate a Self-Signed Certificate</a>. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Key store password</td>
<td>The keystore password. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Certificate alias</td>
<td>The unique name of the certificate signed by Salesforce. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Client Id</td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see <a href="#">Create a Connected App</a>. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td>Client Secret</td>
<td>The OAuth Consumer Secret, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see <a href="#">Create a Connected App</a>. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td><strong>Callback Host</strong></td>
<td>The host value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the API (Enable OAuth Settings) area of the connected app detail page in Salesforce. This property is available only when the OAuth <strong>Implicit Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Callback Port</strong></td>
<td>The port value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the API (Enable OAuth Settings) area of the connected app detail page in Salesforce. This property is available only when the OAuth <strong>Implicit Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Token File</strong></td>
<td>The path to the token file that stores the refresh token used to get the access token without authorization. This property is available only when the OAuth <strong>Implicit Flow</strong> type is selected.</td>
</tr>
</tbody>
</table>

### Advanced settings

| **Salesforce URL** | The Webservice URL required to connect to Salesforce. |
| **API version** | The Salesforce API version. This property is available only when the OAuth connection type is selected. |
| **Bulk Connection** | Select this check box if you need to use bulk data processing function. |
| **Use or save the connection session** | Select this check box and in the **Session directory** field displayed, specify the path to the connection session file to be saved or used. This session file can be shared by different Jobs to retrieve a connection session as long as the correct user ID is provided by the component. This way, you do not need to connect to the server to retrieve the session. When an expired session is detected, if the correct connection information (the user ID, password, and security key) is provided, the component will connect to the server to retrieve the new session information and update the connection session file. This property is available only when the **Basic** connection type is selected. |
| **Need compression** | Select this check box to activate SOAP message compression, which can result in increased performance levels. |
| **Use Http Chunked** | Select this check box to use the HTTP chunked data transfer mechanism. This property is available only when the **Bulk Connection** check box is cleared. |
**tSalesforceConnection**

<table>
<thead>
<tr>
<th><strong>Trace HTTP message</strong></th>
<th>Select this check box to output the HTTP interactions on the console. This property is available only when the <strong>Bulk Connection</strong> check box is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client Id</strong></td>
<td>Enter the ID of the real user to differentiate between those who use the same account and password to access the Salesforce website.</td>
</tr>
<tr>
<td><strong>Timeout</strong></td>
<td>Enter the intended number of query timeout in milliseconds in Salesforce.</td>
</tr>
<tr>
<td><strong>Use Proxy</strong></td>
<td>Select this check box to use a proxy server, and in the <strong>Host</strong>, <strong>Port</strong>, <strong>User Id</strong>, and <strong>Password</strong> fields displayed, specify the connection parameters of the proxy server.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

| **ERROR_MESSAGE** | The error message generated by the component when an error occurs. This is an After variable and it returns a string.                                                                                     |

**Usage**

| **Usage rule** | This component is more commonly used with other Salesforce components.                                                                                                                             |

**Connecting to Salesforce using OAuth implicit flow to authenticate the user (deprecated)**

In this scenario, the **OAuth2** (Open Authorization 2.0) method is adopted to authorize **tSalesforceConnection** to access the Salesforce resources. This way, better security is achieved as users do not need entering their Salesforce username/password directly on **tSalesforceConnection**, which is required by the **Basic** login type instead.
Setting up the Job

Procedure
1. Drop a tSalesforceConnection component, a tFixedFlowInput component and a tSalesforceOutput component onto the workspace.
2. Link the tFixedFlowInput component to the tSalesforceOutput component using a Row > Main connection.
3. Link the tSalesforceConnection component to the tFixedFlowInput component using a Trigger > OnSubjobOk connection.

Configuring the components

Procedure
1. Double-click the tSalesforceConnection component to open its Basic settings view.
2. Select OAuth from the Connection type drop-down list and Implicit Flow from the Oauth2 flow type drop-down list.
3. In the Client Id and Client Secret fields, enter the key and secret used by the consumer to access Salesforce respectively.
4. In the Callback Host and Callback Port fields, enter the callback host and port information respectively.
5. In the Token File field, browse to or enter the path to the token file that stores the refresh token used by the OAuth-enabled connected app to obtain new sessions without requiring the user to provide their credentials.
6. Double-click the tFixedFlowInput component to open its Basic settings view.
7. Click the [...] button next to Edit schema and in the pop-up schema dialog box, define the schema by adding one column Name of String type. When done, click OK to save the changes and close the dialog box.
8. In the **Mode** area, select **Use Inline Content (delimited file)**. Then in the **Content** field displayed, enter the data to be written into Salesforce. For example, **Talend**.

9. Double-click the **tSalesforceOutput** component to open its **Basic settings** view.

10. Select the **tSalesforceConnection** component from the **Connection Component** drop-down list to reuse the connection created by it.

11. Click the [...] button next to the **Module Name** field and in the pop-up dialog box, select the object you want to access. In this example, it is **Account**.

12. Click the [...] button next to **Edit schema** to open the schema dialog box. On the right side of the panel, remove all the columns except the column **Name**. When done, click **OK** to save the changes and close the dialog box.

---

**Executing the Job**

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to run the Job. The Studio console gives the URL (in yellow) for OAuth authorization.
3. Copy the URL to the browser’s address bar. The Salesforce.com login page appears.

4. Fill up the username and password and click **Log in to Salesforce**.
   The authorization response is then shown on the browser at the callback host.

5. Now is the time for the Job to continue running.
As shown above, the Job is successfully executed.

6. Go to the Salesforce.com and check the Account object. You can find that the account whose name is Talend is inserted.

![Image of New Accounts with Talend account inserted]

**Related scenario**

**Upserting Salesforce data based on external IDs** on page 3420
**tSalesforceGetDeleted**

Collects data deleted during a specific period of time from a Salesforce object.

**tSalesforceGetDeleted Standard properties**

These properties are used to configure tSalesforceGetDeleted running in the Standard Job framework. The Standard tSalesforceGetDeleted component belongs to the Business and the Cloud families. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the <strong>Connection Component</strong> drop-down list.</td>
</tr>
<tr>
<td>Connection Component</td>
<td>Select the component that opens the database connection to be reused by this component.</td>
</tr>
<tr>
<td>Connection type</td>
<td>Select the type of the connection from the drop-down list. The connection properties will vary according to the connection type you choose.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Basic</strong>: select this option to access Salesforce by entering your Salesforce username and password.</td>
</tr>
<tr>
<td></td>
<td>• <strong>OAuth</strong>: select this option to access Salesforce using OAuth (Open Authorization) 2.0 protocol to authenticate the user. From the <strong>OAuth2 flow type</strong> drop-down list displayed, you need to choose an OAuth 2.0 authentication flow type, either <strong>Json Web Token Flow</strong> or <strong>Implicit Flow (Deprecated)</strong> (User Agent Flow). This way, you need to create a connected app in Salesforce to configure a Salesforce authentication provider.</td>
</tr>
<tr>
<td></td>
<td>For more information about the OAuth authentication flow, see Authenticate Apps with OAuth.</td>
</tr>
<tr>
<td></td>
<td>For more information about how to create a connected app in Salesforce, see Create a Connected App.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>User Id</td>
<td>The Salesforce username. This property is available only when the Basic connection type is selected.</td>
</tr>
<tr>
<td>Password</td>
<td>The Salesforce password associated with the username. This property is available only when the Basic connection type is selected.</td>
</tr>
<tr>
<td>Security Token</td>
<td>The Salesforce security token. For more information, see Reset Your Security Token. This property is available only when the Basic connection type is selected.</td>
</tr>
<tr>
<td>Issuer</td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see Create a Connected App. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Subject</td>
<td>The Salesforce username. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Expiration time (in seconds)</td>
<td>The expiration time of the assertion (in seconds) within five minutes. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Key store</td>
<td>The path to the keystore file in Java Keystore (JKS) format. The keystore file can be generated by creating a certificate signed by Salesforce and then exporting it to keystore. For more information, see Generate a Self-Signed Certificate. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Key store password</td>
<td>The keystore password. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Certificate alias</td>
<td>The unique name of the certificate signed by Salesforce. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Client Id</td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see Create a Connected App. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td><strong>Client Secret</strong></td>
<td>The OAuth Consumer Secret, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see Create a Connected App. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td><strong>Callback Host</strong></td>
<td>The host value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the API (Enable OAuth Settings) area of the connected app detail page in Salesforce. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td><strong>Callback Port</strong></td>
<td>The port value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the API (Enable OAuth Settings) area of the connected app detail page in Salesforce. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td><strong>Token File</strong></td>
<td>The path to the token file that stores the refresh token used to get the access token without authorization. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td><strong>Module Name</strong></td>
<td>Click the [...] button next to the field and in the dialog box displayed, select the module that will be used or select the Use custom object check box and specify the module name in the Object Name field.</td>
</tr>
</tbody>
</table>
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  • View schema: choose this option to view the schema only.  
  • Change to built-in property: choose this option to change the schema to Built-in for local changes.  
  • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. Click Sync columns to retrieve the schema from the previous component connected in the Job. |
### Start Date
Click the [...] button next to the field and in the calendar dialog box displayed, select the date and time to collect only the data deleted since the specified time. Note that you can collect only the data deleted in the past 30 days.

### End Date
Click the [...] button next to the field and in the calendar dialog box displayed, select the date and time to collect only the data deleted before the specified time.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salesforce URL</td>
<td>The Webservice URL required to connect to Salesforce.</td>
</tr>
<tr>
<td>API version</td>
<td>The Salesforce API version. This property is available only when the OAuth connection type is selected.</td>
</tr>
<tr>
<td>Need compression</td>
<td>Select this check box to activate SOAP message compression, which can result in increased performance levels.</td>
</tr>
<tr>
<td>Use Http Chunked</td>
<td>Select this check box to use the HTTP chunked data transfer mechanism.</td>
</tr>
<tr>
<td>Client Id</td>
<td>Enter the ID of the real user to differentiate between those who use the same account and password to access the Salesforce website.</td>
</tr>
<tr>
<td>Timeout</td>
<td>Enter the intended number of query timeout in milliseconds in Salesforce.</td>
</tr>
<tr>
<td>Use Proxy</td>
<td>Select this check box to use a proxy server, and in the Host, Port, User Id, and Password fields displayed, specify the connection parameters of the proxy server.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This component is usually used as a start component of a Job or Subjob and it always needs an output link.</td>
<td></td>
</tr>
</tbody>
</table>
Recovering deleted data from Salesforce

This scenario describes a two-component Job that collects the account data deleted over the past five days from Salesforce.

Setting up the Job for recovering deleted data from Salesforce

Procedure

1. Create a new Job and add a `tSalesforceGetDeleted` component and a `tLogRow` component by typing their names on the design workspace or dropping them from the Palette.
2. Connect the `tSalesforceGetDeleted` component to the `tLogRow` component using a Row > Main connection.

Configuring the Job for recovering deleted data from Salesforce

Procedure

1. Double-click the `tSalesforceGetDeleted` component to open its Basic settings view.
2. In the User Id, Password and Security Key fields, enter the user authentication information required to access Salesforce.
3. Click the [...] button next to the Module Name field and in the pop-up dialog box, select the object you want to access. In this example, it is Account. When done, click OK to save the settings and close the dialog box.
4. Click the [...] button next to the Start Date field and in the pop-up calendar dialog box, set the value of the start date and time to collect only the data deleted since the specified time. In this example, it is 2016-06-08 00:00:00.

5. Do the same to set the value of the end date and time in the End Date field to collect only the data deleted before the specified time. In this example, it is 2016-06-13 00:00:00.

6. Double-click the tLogRow component to open its Basic settings view.
7. Click **Sync columns** to retrieve the schema from the preceding component.
8. In the **Mode** area, select **Vertical (each row is a key/value list)** to display the results in a tabular form on the console.

**Executing the Job to recover deleted data from Salesforce**

**Procedure**

1. Press **Ctrl + S** to save your Job.
2. Press **F6** to run your Job.
As shown above, the data deleted during the past five days is collected and displayed in a tabular form on the console.
tSalesforceGetServerTimestamp

Retrieves the current date of the Salesforce server presented in a timestamp format.

**tSalesforceGetServerTimestamp Standard properties**

These properties are used to configure tSalesforceGetServerTimestamp running in the Standard Job framework.

The Standard tSalesforceGetServerTimestamp component belongs to the Business and the Cloud families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

This property is not available when other connection component is selected from the Connection Component drop-down list.

| Connection Component | Select the component that opens the database connection to be reused by this component. |

<table>
<thead>
<tr>
<th>Connection type</th>
<th>Select the type of the connection from the drop-down list. The connection properties will vary according to the connection type you choose.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Basic:</strong> select this option to access Salesforce by entering your Salesforce username and password.</td>
</tr>
<tr>
<td></td>
<td>• <strong>OAuth:</strong> select this option to access Salesforce using OAuth (Open Authorization) 2.0 protocol to authenticate the user. From the OAuth2 flow type drop-down list displayed, you need to choose an OAuth 2.0 authentication flow type, either Json Web Token Flow or Implicit Flow (Deprecated) (User Agent Flow). This way, you need to create a connected app in Salesforce to configure a Salesforce authentication provider.</td>
</tr>
</tbody>
</table>

For more information about the OAuth authentication flow, see Authenticate Apps with OAuth.
For more information about how to create a connected app in Salesforce, see [Create a Connected App](#).

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User Id</strong></td>
<td>The Salesforce username.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the <strong>Basic</strong> connection type is selected.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>The Salesforce password associated with the username.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the <strong>Basic</strong> connection type is selected.</td>
</tr>
<tr>
<td><strong>Security Token</strong></td>
<td>The Salesforce security token. For more information, see <a href="#">Reset Your Security Token</a>.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the <strong>Basic</strong> connection type is selected.</td>
</tr>
<tr>
<td><strong>Issuer</strong></td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see <a href="#">Create a Connected App</a>.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Subject</strong></td>
<td>The Salesforce username.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Expiration time (in seconds)</strong></td>
<td>The expiration time of the assertion (in seconds) within five minutes.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Key store</strong></td>
<td>The path to the keystore file in Java Keystore (JKS) format.</td>
</tr>
<tr>
<td></td>
<td>The keystore file can be generated by creating a certificate signed by Salesforce and then exporting it to keystore. For more information, see <a href="#">Generate a Self-Signed Certificate</a>.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Key store password</strong></td>
<td>The keystore password.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Certificate alias</strong></td>
<td>The unique name of the certificate signed by Salesforce.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Client Id</strong></td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see <a href="#">Create a Connected App</a>.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Client Secret</td>
<td>The OAuth Consumer Secret, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see Create a Connected App.</td>
</tr>
<tr>
<td>Callback Host</td>
<td>The host value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the API (Enable OAuth Settings) area of the connected app detail page in Salesforce.</td>
</tr>
<tr>
<td>Callback Port</td>
<td>The port value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the API (Enable OAuth Settings) area of the connected app detail page in Salesforce.</td>
</tr>
<tr>
<td>Token File</td>
<td>The path to the token file that stores the refresh token used to get the access token without authorization.</td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salesforce URL</td>
<td>The Webservice URL required to connect to Salesforce.</td>
</tr>
<tr>
<td>API version</td>
<td>The Salesforce API version.</td>
</tr>
</tbody>
</table>
**Use or save the connection session**

Select this check box and in the **Session directory** field displayed, specify the path to the connection session file to be saved or used.

This session file can be shared by different Jobs to retrieve a connection session as long as the correct user ID is provided by the component. This way, you do not need to connect to the server to retrieve the session.

When an expired session is detected, if the correct connection information (the user ID, password, and security key) is provided, the component will connect to the server to retrieve the new session information and update the connection session file.

This property is available only when the **OAuth** connection type is selected.

**Need compression**

Select this check box to activate SOAP message compression, which can result in increased performance levels.

**Use Http Chunked**

Select this check box to use the HTTP chunked data transfer mechanism.

**Client Id**

Enter the ID of the real user to differentiate between those who use the same account and password to access the Salesforce website.

**Timeout**

Enter the intended number of query timeout in milliseconds in Salesforce.

**Use Proxy**

Select this check box to use a proxy server, and in the **Host**, **Port**, **User Id**, and **Password** fields displayed, specify the connection parameters of the proxy server.

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

<table>
<thead>
<tr>
<th>Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong></td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

**Usage**

**Usage rule**

This component is usually used as a start component of a Job or Subjob and it always needs an output link.
Related scenario

No scenario is available for this component yet.
tSalesforceGetUpdated

Collects data updated during a specific period of time from a Salesforce object.

**tSalesforceGetUpdated Standard properties**

These properties are used to configure tSalesforceGetUpdated running in the Standard Job framework.
The Standard tSalesforceGetUpdated component belongs to the Business and the Cloud families.
The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up <strong>Repository Content</strong> dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in. This property is not available when other connection component is selected from the <strong>Connection Component</strong> drop-down list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component that opens the database connection to be reused by this component.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Connection type</th>
<th>Select the type of the connection from the drop-down list. The connection properties will vary according to the connection type you choose.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Basic</strong>: select this option to access Salesforce by entering your Salesforce username and password.</td>
</tr>
<tr>
<td></td>
<td>• <strong>OAuth</strong>: select this option to access Salesforce using OAuth (Open Authorization) 2.0 protocol to authenticate the user. From the OAuth2 flow type drop-down list displayed, you need to choose an OAuth 2.0 authentication flow type, either <strong>Json Web Token Flow</strong> or <strong>Implicit Flow (Deprecated)</strong> (User Agent Flow). This way, you need to create a connected app in Salesforce to configure a Salesforce authentication provider. For more information about how to create a connected app in Salesforce, see <strong>Create a Connected App</strong>. For more information about the OAuth authentication flow, see <strong>Authenticate Apps with OAuth</strong>.</td>
</tr>
</tbody>
</table>

For more information about how to create a connected app in Salesforce, see Create a Connected App.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Id</td>
<td>The Salesforce username. This property is available only when the Basic connection type is selected.</td>
</tr>
<tr>
<td>Password</td>
<td>The Salesforce password associated with the username. This property is available only when the Basic connection type is selected.</td>
</tr>
<tr>
<td>Security Token</td>
<td>The Salesforce security token. For more information, see Reset Your Security Token. This property is available only when the Basic connection type is selected.</td>
</tr>
<tr>
<td>Issuer</td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see Create a Connected App. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Subject</td>
<td>The Salesforce username. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Expiration time (in seconds)</td>
<td>The expiration time of the assertion (in seconds) within five minutes. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Key store</td>
<td>The path to the keystore file in Java Keystore (JKS) format. The keystore file can be generated by creating a certificate signed by Salesforce and then exporting it to keystore. For more information, see Generate a Self-Signed Certificate. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Key store password</td>
<td>The keystore password. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Certificate alias</td>
<td>The unique name of the certificate signed by Salesforce. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td>Client Id</td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see Create a Connected App. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
</tbody>
</table>
| **Client Secret** | The OAuth Consumer Secret, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see [Create a Connected App](#).

This property is available only when the OAuth **Implicit Flow** type is selected. |
| **Callback Host** | The host value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the [API (Enable OAuth Settings)](#) area of the connected app detail page in Salesforce.

This property is available only when the OAuth **Implicit Flow** type is selected. |
| **Callback Port** | The port value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the [API (Enable OAuth Settings)](#) area of the connected app detail page in Salesforce.

This property is available only when the OAuth **Implicit Flow** type is selected. |
| **Token File** | The path to the token file that stores the refresh token used to get the access token without authorization.

This property is available only when the OAuth **Implicit Flow** type is selected. |
| **Module Name** | Click the [...] button next to the field and in the dialog box displayed, select the module that will be used or select the [Use custom object](#) check box and specify the module name in the **Object Name** field. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click [Edit schema](#) to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content](#) window.

Click [Sync columns](#) to retrieve the schema from the previous component connected in the Job. |
### Start Date

Click the [...] button next to the field and in the calendar dialog box displayed, select the date and time to collect only the data updated since the specified time. Note that you can collect only the data updated in the past 30 days.

### End Date

Click the [...] button next to the field and in the calendar dialog box displayed, select the date and time to collect only the data updated before the specified time.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Salesforce URL</strong></th>
<th>The Webservice URL required to connect to Salesforce.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API version</strong></td>
<td>The Salesforce API version. This property is available only when the OAuth connection type is selected.</td>
</tr>
<tr>
<td><strong>Use or save the connection session</strong></td>
<td>Select this check box and in the Session directory field displayed, specify the path to the connection session file to be saved or used. This session file can be shared by different Jobs to retrieve a connection session as long as the correct user ID is provided by the component. This way, you do not need to connect to the server to retrieve the session. When an expired session is detected, if the correct connection information (the user ID, password, and security key) is provided, the component will connect to the server to retrieve the new session information and update the connection session file. This property is available only when the Basic connection type is selected.</td>
</tr>
<tr>
<td><strong>Need compression</strong></td>
<td>Select this check box to activate SOAP message compression, which can result in increased performance levels.</td>
</tr>
<tr>
<td><strong>Use Http Chunked</strong></td>
<td>Select this check box to use the HTTP chunked data transfer mechanism.</td>
</tr>
<tr>
<td><strong>Client Id</strong></td>
<td>Enter the ID of the real user to differentiate between those who use the same account and password to access the Salesforce website.</td>
</tr>
<tr>
<td><strong>Timeout</strong></td>
<td>Enter the intended number of query timeout in milliseconds in Salesforce.</td>
</tr>
<tr>
<td><strong>Use Proxy</strong></td>
<td>Select this check box to use a proxy server, and in the Host, Port, User Id, and Password fields displayed, specify the connection parameters of the proxy server.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>
Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This component is usually used as a start component of a Job or Subjob and it always needs an output link.</td>
</tr>
</tbody>
</table>

Related scenario

No scenario is available for this component yet.
tSalesforceInput

Retrieves data from a Salesforce object based on a query.

**tSalesforceInput Standard properties**

These properties are used to configure tSalesforceInput running in the Standard Job framework.
The Standard tSalesforceInput component belongs to the Business and the Cloud families.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component that opens the database connection to be reused by this component.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Connection type</th>
<th>Select the type of the connection from the drop-down list. The connection properties will vary according to the connection type you choose.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Basic:</strong> select this option to access Salesforce by entering your Salesforce username and password.</td>
</tr>
<tr>
<td></td>
<td>• <strong>OAuth:</strong> select this option to access Salesforce using OAuth (Open Authorization) 2.0 protocol to authenticate the user. From the OAuth2 flow type drop-down list displayed, you need to choose an OAuth 2.0 authentication flow type, either Json Web Token Flow or Implicit Flow (Deprecated) (User Agent Flow). This way, you need to create a connected app in Salesforce to configure a Salesforce authentication provider. For more information about the OAuth authentication flow, see Authenticate Apps with OAuth. For more information about how to create a connected app in Salesforce, see Create a Connected App.</td>
</tr>
<tr>
<td><strong>User Id</strong></td>
<td>The Salesforce username. This property is available only when the Basic connection type is selected.</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>The Salesforce password associated with the username. This property is available only when the Basic connection type is selected.</td>
</tr>
<tr>
<td><strong>Security Token</strong></td>
<td>The Salesforce security token. For more information, see Reset Your Security Token. This property is available only when the Basic connection type is selected.</td>
</tr>
<tr>
<td><strong>Issuer</strong></td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see Create a Connected App. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td><strong>Subject</strong></td>
<td>The Salesforce username. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td><strong>Expiration time (in seconds)</strong></td>
<td>The expiration time of the assertion (in seconds) within five minutes. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td><strong>Key store</strong></td>
<td>The path to the keystore file in Java Keystore (JKS) format. The keystore file can be generated by creating a certificate signed by Salesforce and then exporting it to keystore. For more information, see Generate a Self-Signed Certificate. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td><strong>Key store password</strong></td>
<td>The keystore password. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td><strong>Certificate alias</strong></td>
<td>The unique name of the certificate signed by Salesforce. This property is available only when the OAuth Json Web Token Flow type is selected.</td>
</tr>
<tr>
<td><strong>Client Id</strong></td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see Create a Connected App. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
</tbody>
</table>
| **Client Secret** | The OAuth Consumer Secret, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see [Create a Connected App](#).  
This property is available only when the OAuth Implicit Flow type is selected. |
| **Callback Host** | The host value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the API (Enable OAuth Settings) area of the connected app detail page in Salesforce.  
This property is available only when the OAuth Implicit Flow type is selected. |
| **Callback Port** | The port value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the API (Enable OAuth Settings) area of the connected app detail page in Salesforce.  
This property is available only when the OAuth Implicit Flow type is selected. |
| **Token File** | The path to the token file that stores the refresh token used to get the access token without authorization.  
This property is available only when the OAuth Implicit Flow type is selected. |
| **Module Name** | Click the [...] button next to the field and in the dialog box displayed, select the module that will be used or select the Use custom object check box and specify the module name in the Object Name field. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
- View schema: choose this option to view the schema only.  
- Change to built-in property: choose this option to change the schema to Built-in for local changes.  
- Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.  
In this component the schema is related to the Module selected.  
To retrieve a column from a linked module it is necessary to define the column in a particular manner in the Edit schema view, otherwise the relationship query

will not work. The correct syntax is: `NameofCurrentModule_NameofLinkedModule_NameofColumnofInterest`.

| Query Mode | Select the query mode from the drop-down list, either **Query** or **Bulk**.  
|---|---|
| Query | the ordinary query.  
| Bulk | the bulk query used to query large data sets efficiently and reduce the number of API requests.  

| Condition | Enter the query used to select the data to be extracted between double quotation marks, for example, "name='Talend'" or "name like '%talend_user%'".  
This property is not available when the Manual Query check box is selected.  

| Manual Query | Select this check box and in the **Full SOQL query string** field displayed, enter the full SOQL (Salesforce Object Query Language) statement used to select the data to be retrieved between double quotation marks. For more information about the SOQL, see Salesforce Object Query Language (SOQL).  
This property is not available when the Manual Query check box is selected.  

| Guess schema | Click this button to generate the schema columns based on the query specified in the **Full SOQL query string** field.  
This property is available only when the Manual Query check box is selected.  
Note that there are some limitations you should be aware of when using this feature. Please refer to the Know Limitations section in Release Notes for more detailed information.  

| Guess query | Click this button to generate the query in the **Full SOQL query string** field based on the defined module and schema.  
For more information about how to set the module and schema correctly when generating the SOQL queries, see How to set schema for the guess query feature of tSalesforceInput on page 3410.  
This property is available only when the Manual Query check box is selected.  
Note that there are some limitations you should be aware of when using this feature. Please refer to the Know Limitations section in Release Notes for more detailed information.  

| Include deleted records | Select this check box to query all the records, including the deleted ones.  
This property is available only when **Query** is selected from the **Query Mode** drop-down list.  

## Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salesforce URL</td>
<td>The Webservice URL required to connect to Salesforce.</td>
</tr>
<tr>
<td>API version</td>
<td>The Salesforce API version. This property is available only when the OAuth connection type is selected.</td>
</tr>
<tr>
<td>Use or save the connection session</td>
<td>Select this box and in the Session directory field displayed, specify the path to the connection session file to be saved or used.</td>
</tr>
<tr>
<td></td>
<td>This session file can be shared by different Jobs to retrieve a connection session as long as the correct user ID is provided by the component. This way, you do not need to connect to the server to retrieve the session.</td>
</tr>
<tr>
<td></td>
<td>When an expired session is detected, if the correct connection information (the user ID, password, and security key) is provided, the component will connect to the server to retrieve the new session information and update the connection session file.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when the Basic connection type is selected.</td>
</tr>
<tr>
<td>Need compression</td>
<td>Select this box to activate SOAP message compression, which can result in increased performance levels.</td>
</tr>
<tr>
<td>Trace HTTP message</td>
<td>Select this box to output the HTTP interactions on the console.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when Bulk is selected from the Query Mode drop-down list.</td>
</tr>
<tr>
<td>Use HTTP Chunked</td>
<td>Select this box to use the HTTP chunked data transfer mechanism.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when Query is selected from the Query Mode drop-down list.</td>
</tr>
<tr>
<td>Client Id</td>
<td>Enter the ID of the real user to differentiate between those who use the same account and password to access the Salesforce website.</td>
</tr>
<tr>
<td>Timeout</td>
<td>Enter the intended number of query timeout in milliseconds in Salesforce.</td>
</tr>
<tr>
<td>Use Proxy</td>
<td>Select this box to use a proxy server, and in the Host, Port, User Id, and Password fields displayed, specify the connection parameters of the proxy server.</td>
</tr>
<tr>
<td>Batch Size</td>
<td>Enter the number of registrations in each processed batch.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when Query is selected from the Query Mode drop-down list.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Normalize Delimiter</td>
<td>Enter the characters, strings or regular expressions used to normalize the data that is collected by queries set on different hierarchical Salesforce objects. This property is available only when <strong>Query</strong> is selected from the <strong>Query Mode</strong> drop-down list.</td>
</tr>
<tr>
<td>Column Name Delimiter</td>
<td>Enter the characters, strings or regular expressions used to separate the name of the parent object from the name of the child object when you use a query on the hierarchical relations among the different Salesforce objects. This property is available only when <strong>Query</strong> is selected from the <strong>Query Mode</strong> drop-down list.</td>
</tr>
<tr>
<td>Safety Switch</td>
<td>Clear this check box if the length of any column of the module to be queried is greater than 100,000 characters. By default, this check box is selected to prevent excessive memory usage. This property is available only when <strong>Bulk</strong> is selected from the <strong>Query Mode</strong> drop-down list.</td>
</tr>
<tr>
<td>Complete Job timeout</td>
<td>The timeout value, in seconds, within which your bulk query Job must be completed. By default, the value is set to 0, which means there is no time limit for your Job to complete. Salesforce has its own limit for bulk query. If it takes time longer than Salesforce's timeout limit to process a batch, the batch will be placed in a queue and processed later, and thus it might take too much time for the Job to execute. In this case, you can define the Complete Job timeout value to make your Job end in advance. For more information about Salesforce's own limit, see Bulk API Limits - Batch processing time. This property is available only when <strong>Bulk</strong> is selected from the <strong>Query Mode</strong> drop-down list.</td>
</tr>
<tr>
<td>Enable PK Chunking</td>
<td>Select this check box to enable PK (Primary Key, i.e., the object's record ID) chunking when extracting large amounts of records or when the query consistently times out. In the <strong>Chunk size</strong> field displayed, specify the number of records within the ID boundaries for each chunk. For more information, see Use PK Chunking to Extract Large Data Sets from Salesforce. This property is available only when <strong>Bulk</strong> is selected from the <strong>Query Mode</strong> drop-down list.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
<tr>
<td><strong>Global Variables</strong></td>
<td></td>
</tr>
<tr>
<td><strong>NB_LINE</strong></td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>
The error message generated by the component when an error occurs. This is an After variable and it returns a string.

**Usage**

| Usage rule | This component is usually used as a start component of a Job or Subjob and it always needs an output link. |

### How to set schema for the guess query feature of tSalesforceInput

**tSalesforceInput** allows you to generate SOQL queries, with both standard objects and fields and custom objects and fields, based on the defined module and schema. This section demonstrates how to set the module and schema when generating an SOQL query.

There are two types of SOQL queries: simple query and relationship query.

Simple query retrieves data from only one object. For more information about how to set the module and schema for generating simple queries, see **How to set schema for generating the simple query** on page 3410.

Relationship query retrieves data from more than a single type of object, including child-to-parent and parent-to-child relationship queries.

For more information about how to set the module and schema for generating child-to-parent relationship queries, see **How to set schema for generating the child-to-parent relationship query** on page 3411.

For more information about how to set the module and schema for generating parent-to-child relationship queries, see **How to set schema for generating the parent-to-child relationship query** on page 3412.

Note that before setting the module name and schema columns for generating the relationship query in the **Basic settings** view of **tSalesforceInput**, you need to first identify the relationship type (child-to-parent or parent-to-child) of the query to be generated, because the methods of setting the schema column names for these two kinds of relationship queries are somewhat different. The main difference between the child-to-parent and parent-to-child relationship queries is that the parent-to-child relationship query is specified by using the subquery enclosed in parentheses, while the child-to-parent relationship query is not. For more information about the relationship queries in SOQL and how to identify the relationship queries, see **Relationship Queries**.

### How to set schema for generating the simple query

This section demonstrates how to set the module name and schema columns for generating the simple query.

The following two simple query examples will be used for demonstration purpose in the following steps.

- SELECT Id, Name, BillingCity FROM Account, a simple query with standard object and fields, and
- SELECT Name__c, LastName__c FROM Mother__c, a simple query with custom object and fields.
Procedure

1. Set the module name with the name of the object specified in the FROM clause, Account and Mother__c for above examples.

2. Create a column for each field in the field list (separated by commas) after SELECT in the schema dialog box and set the column name with the field name.

   - For the first example, you need to create three columns Id, Name, and BillingCity for the three fields.
   - For the second example, you need to create two columns Name__c and LastName__c for the two fields.

How to set schema for generating the child-to-parent relationship query

This section demonstrates how to set the module name and schema columns for generating a child-to-parent relationship query.

The following two child-to-parent relationship query examples will be used for demonstration purpose in the following steps.

- SELECT Name, Account.Name, Account.Owner.Name FROM Contact, a child-to-parent relationship query with standard object and fields, and

- SELECT Id, FirstName__c, MotherOfDaughter__r.FirstName__c FROM Daughter__c, a child-to-parent relationship query with custom object and fields.

   Note that here you must use the relationship name with __r instead of __c. For more information, see Understanding Relationship Names, Custom Objects, and Custom Fields.

Procedure

1. Set the module name with the name of the object specified in the FROM clause, Contact and Daughter__c in above examples.

2. Create a column for each field in the field list (separated by commas) after SELECT in the schema dialog box.

   - For the first example, you need to create three columns for the three fields Name, Account.Name, and Account.Owner.Name.
   - For the second example, you need to create three columns for the three fields Id, FirstName__c, and MotherOfDaughter__r.FirstName__c.

3. Set the name of each column with the name of each field and replace all dots in the column name with underscore characters.

   - For the first example, the names of the three columns are set to Name, Account_Name, and Account_Owner_Name.
   - For the second example, the names of the three columns are set to Id, FirstName__c, and MotherOfDaughter__r_FirstName__c.

4. Set the type of each column.

   The schema for the first example should be set like this:
And the schema for the second example should be set like this:

Note that the underscore character '_-' is used as a separator between the relationship name and the field name in Talend schema, so only the underscore character '_' that goes after '__r' or '__c' will be replaced by the dot character '.' when generating the query. If the underscore character '_' is a part of any custom name in the schema, for example, `Contact_custom_field__c`, which should be `Contact.custom_field__c` in the query, you need to replace '_' in the generated query with '.' manually.

**How to set schema for generating the parent-to-child relationship query**

This section demonstrates how to set the module name and schema columns for generating a parent-to-child relationship query.

The following two parent-to-child relationship query examples will be used for demonstration purpose in the following steps.

- SELECT Name, Owner.Name (SELECT CreatedBy.Name FROM Notes) FROM Account, a parent-to-child relationship query with standard object and fields, and
- SELECT LastName__c, (SELECT FirstName__c FROM Daughters__r) FROM Mother__c, a parent-to-child relationship query with custom object and fields.
Note that here you must use the relationship name with __r instead of __c. For more information, see Understanding Relationship Names, Custom Objects, and Custom Fields.

Procedure

1. Set the module name with the name of the object specified in the outer query FROM clause, Account and Mother__c in above examples.

2. Create a column for each field (including the fields in subquery) after SELECT in the schema dialog box.

   For the first example, you need to create three columns for the three fields Name and Owner.Name after the outer SELECT, and one field CreatedBy.Name after the subquery SELECT.

   For the second example, you need to create two columns for the two fields LastName__c after the outer SELECT, and the field FirstName__c after the subquery SELECT.

3. For the fields in the outer SELECT clause, which are outside parentheses, set the name of each column with the name of each field and replace all dots in the column name with underscore characters.

   For the first example, the column names for the two fields Name and Owner.Name in the outer SELECT clause are set to Name and Owner_Name.

   For the second example, there is no dot in the field name, so the column name is same as the field name.

4. For the fields in the subquery SELECT, construct the column names using the pattern <$XXX>_records_<$YYY>, where <$XXX> corresponds to the name of the object specified in the subquery FROM clause, and <$YYY> will be the field name with all dots replaced by underscore characters.

   For the first example, the column name for the field CreatedBy.Name in the subquery is set to Notes_records_CreatedBy_Name.

   For the second example, the column name for the field FirstName__c in the subquery is set to Daughters__r_records_FirstName__c.

5. Set the type of each column.

The schema for the first example should be set like this:
And the schema for the second example should be set like this:

![Schema of tSalesforceInput](image)

Note that the underscore character '_' is used as a separator between the relationship name and the field name in Talend schema, so only the underscore character '_' that goes after '__r' or '__c' will be replaced by the dot character '.' when generating the query. If the underscore character '_' is a part of any custom name in the schema, for example, Contact_custom_field__c, which should be Contact.custom_field__c in the query, you need to replace '_' in the generated query with '.' manually.

**Related scenario**

Upserting Salesforce data based on external IDs on page 3420
tSalesforceOutput

Inserts, updates, upserts, or deletes data in a Salesforce object.

tSalesforceOutput Standard properties

These properties are used to configure tSalesforceOutput running in the Standard Job framework. The Standard tSalesforceOutput component belongs to the Business and the Cloud families. The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in. This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the component that opens the database connection to be reused by this component.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connection type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the type of the connection from the drop-down list. The connection properties will vary according to the connection type you choose.</td>
<td></td>
</tr>
<tr>
<td>- <strong>Basic</strong>: select this option to access Salesforce by entering your Salesforce username and password.</td>
<td></td>
</tr>
<tr>
<td>- <strong>OAuth</strong>: select this option to access Salesforce using OAuth (Open Authorization) 2.0 protocol to authenticate the user. From the OAuth2 flow type drop-down list displayed, you need to choose an OAuth 2.0 authentication flow type, either Json Web Token Flow or Implicit Flow (Deprecated) (User Agent Flow). This way, you need to create a connected app in Salesforce to configure a Salesforce authentication provider. For more information about the OAuth authentication flow, see Authenticate Apps with OAuth.</td>
<td></td>
</tr>
<tr>
<td>For more information about how to create a connected app in Salesforce, see Create a Connected App.</td>
<td></td>
</tr>
<tr>
<td>User Id</td>
<td>The Salesforce username. This property is available only when the <strong>Basic</strong> connection type is selected.</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Password</td>
<td>The Salesforce password associated with the username. This property is available only when the <strong>Basic</strong> connection type is selected.</td>
</tr>
<tr>
<td>Security Token</td>
<td>The Salesforce security token. For more information, see <a href="#">Reset Your Security Token</a>. This property is available only when the <strong>Basic</strong> connection type is selected.</td>
</tr>
<tr>
<td>Issuer</td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see <a href="#">Create a Connected App</a>. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td>Subject</td>
<td>The Salesforce username. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td>Expiration time (in seconds)</td>
<td>The expiration time of the assertion (in seconds) within five minutes. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td>Key store</td>
<td>The path to the keystore file in Java Keystore (JKS) format. The keystore file can be generated by creating a certificate signed by Salesforce and then exporting it to keystore. For more information, see <a href="#">Generate a Self-Signed Certificate</a>. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td>Key store password</td>
<td>The keystore password. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td>Certificate alias</td>
<td>The unique name of the certificate signed by Salesforce. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td>Client Id</td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see <a href="#">Create a Connected App</a>. This property is available only when the OAuth <strong>Implicit Flow</strong> type is selected.</td>
</tr>
<tr>
<td>Client Secret</td>
<td>The OAuth Consumer Secret, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see Create a Connected App. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Callback Host</td>
<td>The host value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the API (Enable OAuth Settings) area of the connected app detail page in Salesforce. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td>Callback Port</td>
<td>The port value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the API (Enable OAuth Settings) area of the connected app detail page in Salesforce. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td>Token File</td>
<td>The path to the token file that stores the refresh token used to get the access token without authorization. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td>Module Name</td>
<td>Click the [...] button next to the field and in the dialog box displayed, select the module that will be used or select the Use custom object check box and specify the module name in the Object Name field.</td>
</tr>
</tbody>
</table>
| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
- View schema: choose this option to view the schema only.  
- Change to built-in property: choose this option to change the schema to Built-in for local changes.  
- Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.  
Click Sync columns to retrieve the schema from the previous component connected in the Job. |
| Output Action | Select one of the following operations to be performed from the drop-down list. |
- **INSERT**: insert one or more new records into Salesforce.
- **UPDATE**: update one or more existing records in Salesforce.
- **UPSERT**: create new records and update existing records. In the **Upsert Key Column** field displayed, specify the key column for the upsert operation.
- **DELETE**: delete one or more records in Salesforce.

## Advanced settings

<table>
<thead>
<tr>
<th><strong>Salesforce URL</strong></th>
<th>The Webservice URL required to connect to Salesforce.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API version</strong></td>
<td>The Salesforce API version. This property is available only when the <strong>OAuth</strong> connection type is selected.</td>
</tr>
<tr>
<td><strong>Use or save the connection session</strong></td>
<td>Select this check box and in the <strong>Session directory</strong> field displayed, specify the path to the connection session file to be saved or used. This session file can be shared by different Jobs to retrieve a connection session as long as the correct user ID is provided by the component. This way, you do not need to connect to the server to retrieve the session. When an expired session is detected, if the correct connection information (the user ID, password, and security key) is provided, the component will connect to the server to retrieve the new session information and update the connection session file. This property is available only when the <strong>Basic</strong> connection type is selected.</td>
</tr>
<tr>
<td><strong>Need compression</strong></td>
<td>Select this check box to activate SOAP message compression, which can result in increased performance levels.</td>
</tr>
<tr>
<td><strong>Use Http Chunked</strong></td>
<td>Select this check box to use the HTTP chunked data transfer mechanism.</td>
</tr>
<tr>
<td><strong>Client Id</strong></td>
<td>Enter the ID of the real user to differentiate between those who use the same account and password to access the Salesforce website.</td>
</tr>
<tr>
<td><strong>Timeout</strong></td>
<td>Enter the intended number of query timeout in milliseconds in Salesforce.</td>
</tr>
<tr>
<td><strong>Use Proxy</strong></td>
<td>Select this check box to use a proxy server, and in the <strong>Host</strong>, <strong>Port</strong>, <strong>User Id</strong>, and <strong>Password</strong> fields displayed, specify the connection parameters of the proxy server.</td>
</tr>
<tr>
<td><strong>Relationship mapping for upsert</strong></td>
<td>Click the [*] button to add lines as needed and specify the external ID fields in the input flow, the lookup relationship fields in the upsert module, the lookup...</td>
</tr>
</tbody>
</table>
module as well as the external ID fields in the lookup module.

- **Column name of Talend Schema**: the name of the external ID field in the input flow. It refers to the fields in the schema of the preceding component. Such columns are intended to match against the external ID fields specified in the **External id name** column, which are the fields of the lookup module specified in the **Module name** column.

- **Lookup field name**: the name of the lookup field. It refers to the lookup field of the module specified in the **Module Name** field in the **Basic settings** view. This column needs to be specified when there are NULL input values for it and the **Ignore Null** check box is cleared.

- **Lookup relationship field name**: the name of the lookup relationship field. It refers to the lookup relationship fields of the module specified in the **Module Name** field in the **Basic settings** view and is intended to establish relationship with the lookup module specified in the **Module name** column in this table. For how to define the lookup relationship fields and how to provide their correct names in the **Lookup relationship field name** field, go to the Salesforce website and launch the Salesforce Data Loader application for proper actions and information.

- **Module name**: the name of the lookup module.

- **External id name**: the name of the external ID field in the lookup module specified in the **Module name** column.

This property is available only when **UPSERT** is selected from the **Output Action** drop-down list.

<table>
<thead>
<tr>
<th>Extend Insert</th>
<th>Select this check box to transfer the output data in batches. In the <strong>Commit Level</strong> field displayed, specify the number of lines per batch.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cease on Error</td>
<td>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a <strong>Row &gt; Reject</strong> link. The <strong>Extend Insert</strong> check box needs to be cleared when using the <strong>Row &gt; Reject</strong> link to collect the erroneous rows.</td>
</tr>
<tr>
<td>Retrieve Id</td>
<td>Select this check box to allow Salesforce to return the Salesforce ID for the inserted or updated records. This check box is available only when <strong>INSERT</strong> or <strong>UPSERT</strong> is selected from the <strong>Output Action</strong> drop-down list and the <strong>Extend Insert</strong> check box is cleared.</td>
</tr>
</tbody>
</table>
When this check box is selected and the tSalesforceOutput component is linked to another component via a Row > Main connection, the following column(s) will be added to the schema of the data flow:

- **salesforce_id**: the Salesforce ID of the created or updated record.
- **salesforce_upsert_status** (only for UPSERT): the status of the upsert action to indicate whether the record is created or updated.

<table>
<thead>
<tr>
<th>Ignore Null</th>
<th>Select this check box to ignore NULL values. This property is available only when UPDATE or UPSERT is selected from the Output Action drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log File Name</td>
<td>Specify the path to the log file that holds all error logs.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_SUCCESS</td>
<td>The number of rows successfully processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_REJECT</td>
<td>The number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

**Usage**

**Usage rule**

This component can be used as an end component or an intermediate component of a Job or Subjob and it always needs an input link.

**Upserting Salesforce data based on external IDs**

Here is an example of using Talend components to insert and update Salesforce Contact object data based on the mapping relationship with external IDs of the Account object.

The Job used in this example is composed of the following seven subjobs:

- the first subjob opens a connection to Salesforce,
- the second subjob deletes some Account records based on external IDs,
- the third subjob inserts two Account records with external IDs,
• the fourth subjob inserts three Contact records based on the mapping relationship with the Account object external IDs, and gathers both inserted and erroneous data,

• the fifth subjob updates two inserted Contact records based on the mapping relationship with the Account object external IDs,

• the sixth subjob retrieves the inserted Account records, and

• the seventh subjob retrieves the upserted Contact records.
Creating a Job for upserting Salesforce data based on external IDs

Before you begin

- An external ID custom field `Account_External_ID` has been added for the Account object in your Salesforce, which will be used to map the relationship with the Contact object.

- An external ID custom field `Contact_Talend_ID` has been added for the Contact object in Salesforce, which will be used as the key column for upserting Contact object data.

Procedure

1. Create a new Job and add a `tSalesforceConnection` component, three `tSalesforceInput` components, four `tSalesforceOutput` components, three `tFixedFlowInput` components, and four `tLogRow` components.

2. Connect the first `tSalesforceInput` component to the first `tSalesforceOutput` component using a `Row > Main` connection.

3. Do the same to connect the first `tFixedFlowInput` component to the second `tSalesforceOutput` component, the second `tFixedFlowInput` component to the third `tSalesforceOutput` component, the third `tFixedFlowInput` component to the fourth `tSalesforceOutput` component, and the third `tSalesforceInput` component to the fourth `tLogRow` component.

4. Connect the third `tSalesforceOutput` component to the first `tLogRow` component using a `Row > Main` connection and to the second `tLogRow` component using a `Row > Reject` connection.

5. Connect the `tSalesforceConnection` component to the first `tSalesforceInput` component using a `Trigger > On Subjob Ok` connection.

6. Do the same to connect the first `tSalesforceInput` component to the first `tFixedFlowInput` component, the first `tFixedFlowInput` component to the second `tFixedFlowInput` component, the second `tFixedFlowInput` component to the third `tFixedFlowInput` component, the third `tFixedFlowInput` component to the second `tSalesforceInput` component, and the second `tSalesforceInput` component to the third `tSalesforceInput` component.

Opening a connection to Salesforce

Procedure

1. Double-click the `tSalesforceConnection` component to open its `Basic settings` view on the `Component` tab.

2. Select `Basic` from the `Connection type` drop-down list to connect to Salesforce using your Salesforce account username, password, and security token.

   In this example, you connect to Salesforce using the basic connection type. You can also connect to Salesforce using the OAuth JWT (JSON Web Token) Flow type. For more information, see the
related description of how to connect to Salesforce using OAuth JWT Flow on Talend Help Center (https://help.talend.com).

3. In the User Id field, enter your Salesforce username.

4. Click the [...] button next to the Password field and in the pop-up dialog box, enter your Salesforce password between double quotation marks.

5. Click the [...] button next to the Security Token field and in the pop-up dialog box, enter your Salesforce security token between double quotation marks.

You can use the Salesforce metadata wizard to create a connection to your Salesforce system, save the connection in the Repository, and then reuse the connection configuration defined in the metadata wizard in Salesforce components. For more information, see the related description of centralizing Salesforce metadata.

Deleting Salesforce Account records based on external ID

**Procedure**

1. Double-click the first tSalesforceInput component to open its Basic settings view on the Component tab.
2. Select the component that will create the Salesforce connection from the Connection Component drop-down list, tSalesforceConnection_1 in this example.
3. Click the [...] button next to the Module Name field and in the pop-up dialog box, select the object whose data will be deleted, Account in this example. The schema of the Account object will be automatically filled.
4. Click OK to save the changes and in the pop-up dialog box, click Yes to propagate the schema to the next tSalesforceOutput component.
5. Select Query from the Query Mode drop-down list and specify the condition used to filter data to be deleted. In this example, it is talendlena__Account_External_ID__c like '%talend%'.

   All Salesforce Account records whose talendlena__Account_External_ID__c field contains talend will be deleted.

6. Double-click the first tSalesforceOutput component to open its Basic settings view on the Component tab.
7. Select the component that will create the Salesforce connection from the Connection Component drop-down list, tSalesforceConnection_1 in this example.
8. Click the [...] button next to the Module Name field and in the pop-up dialog box, select the object whose data will be deleted, Account in this example.
9. Select DELETE from the Output Action drop-down list to perform the delete operation on the Account objects filtered by the first tSalesforceInput component.

Inserting Salesforce Account records with external IDs

**Procedure**

1. Double-click the second tSalesforceOutput component to open its Basic settings view on the Component tab.
2. Select the component that will create the Salesforce connection from the Connection Component drop-down list, tSalesforceConnection_1 in this example.
3. Click the [...] button next to the **Module Name** field and in the pop-up dialog box, select the object into which data will be inserted, **Account** in this example. The schema of the Account object will be automatically filled.

4. Click the [...] button next to **Edit schema** and in the pop-up dialog box, remove all schema columns except the two columns **Name** and **talendlena__Account__External__ID__c**. Then click ![arrow] to copy these two columns to the first **tFixedFlowInput** component. When done, click **OK** to close the dialog box.

5. Select **INSERT** from the **Output Action** drop-down list.

6. Double-click the first **tFixedFlowInput** component to open its **Basic settings** view on the **Component** tab.

7. In the **Mode** area, select **Use Inline Table** and in the table displayed, enter the data to be inserted into the Salesforce Account object. In this example, the following two records will be inserted.

   ```
   account_talend;account_talend_exid
   account_talend_doc;account_talend_doc_exid
   ```

   You can also select **Use Inline Content**, and then copy and paste the input data into the **Content** field displayed.

### Inserting Salesforce Contact records upon Account external IDs

**Procedure**

1. Double-click the third **tSalesforceOutput** component to open its **Basic settings** view on the **Component** tab.

2. Select the component that will create the Salesforce connection from the **Connection Component** drop-down list, **tSalesforceConnection_1** in this example.

3. Click the [...] button next to the **Module Name** field and in the pop-up dialog box, select the object into which data will be inserted, **Contact** in this example. The schema of the Contact object will be automatically filled. When done, click **OK** to save the changes and in the pop-up dialog box, click **Yes** to propagate the schema to the next two **tLogRow** components.
4. Click the [...] button next to Edit schema and in the pop-up dialog box, remove all schema columns except the three columns LastName, FirstName, and talendlena__Contact_Talend_ID__c. Then add another column Account_External_ID of String type and copy these four columns to the second tFixedFlowInput component. When done, click OK to close the dialog box and in the pop-up dialog box, click Yes to propagate the schema to the next two tLogRow components.

5. Select UPSERT from the Output Action drop-down list and talendlena__Contact_Talend_ID__c from the Upsert Key Column drop-down list displayed.

6. Go to the Advanced settings view and in the Relationship mapping for upsert table, specify the relationship mapping for the column Account_External_ID by adding one row and setting the value for each table column. In this example, the column Account_External_ID is mapped with the external ID field talendlena__Account_External_ID__c of the Account object, so the value for Column name of Talend Schema is Account_External_ID, the value for Lookup relationship field name and Module name is Account, and the value for External id name is talendlena__Account_External_ID__c.

7. Clear the Extend Insert and Die on Error check boxes so that erroneous data can be gathered via the Row > Reject connection.

8. Double-click the second tFixedFlowInput component to open its Basic settings view on the Component tab.
9. In the Mode area, select Use Inline Table and in the table displayed, enter the data to be inserted into the Salesforce Contact object. In this example, the input data is as follows.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Beckham;David;2018010001;account_talend_exid</td>
<td>Taylor;Swift;2018010002;account_talend_exid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>;</td>
<td></td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>;</td>
<td></td>
<td>;</td>
<td></td>
</tr>
</tbody>
</table>

You can also select Use Inline Content, and then copy and paste the input data into the Content field displayed.

10. Double-click the first tLogRow component and on its Basic settings view, select Table in the Mode area to display the result in a table.

11. Do the same to configure the second tLogRow component.

**Updating Salesforce Contact records upon Account external IDs**

**Procedure**

1. Double-click the fourth tSalesforceOutput component to open its Basic settings view on the Component tab.

2. Select the component that will create the Salesforce connection from the Connection Component drop-down list, tSalesforceConnection_1 in this example.

3. Click the [...] button next to the Module Name field and in the pop-up dialog box, select the object into which data will be inserted, Contact in this example. The schema of the Contact object will be automatically filled.

4. Click the [...] button next to Edit schema and in the pop-up dialog box, remove all schema columns except the three columns Phone, Email, and talendlena__Contact_Talend_ID__c. Then add another column Account_External_ID of String type and copy these four columns to the second tFixedFlowInput component. When done, click OK to close the dialog box.

5. Select UPSERT from the Output Action drop-down list and talendlena__Contact_Talend_ID__c from the Upsert Key Column drop-down list displayed.

6. Go to the Advanced settings view and in the Relationship mapping for upsert table, specify the relationship mapping for the column Account_External_ID by adding one row and setting the value for each table column. In this example, the column Account_External_ID is mapped with the external ID field talendlena__Account_External_ID__c of the Account object, so the value for Column name of Talend Schema is Account_External_ID, the value for Lookup relationship field name and Module name is Account, and the value for External id name is talendlena__Account_External_ID__c.

7. Double-click the third tFixedFlowInput component to open its Basic settings view on the Component tab.
8. In the Mode area, select Use Inline Table and in the table displayed, enter the data used to update the two records inserted into the Salesforce Contact object. In this example, the updated data is as follows. The phone and email values will be added for both two records and the Account external ID value for the second record is updated to account_talend_doc_exid.

+86 13666666666;beckham.david@talend.com;2018010001;account_talend_exid
+86 13888888888;taylor.swift@talend.com;2018010002;account_talend_doc_exid

You can also select Use Inline Content, and then copy and paste the input data into the Content field displayed.

Retrieving inserted Salesforce Account records

Procedure

1. Double-click the second tSalesforceInput component to open its Basic settings view on the Component tab.
2. Select the component that will create the Salesforce connection from the Connection Component drop-down list, tSalesforceConnection_1 in this example.
3. Click the [...] button next to the Module Name field and in the pop-up dialog box, select the object from which data will be retrieved, Account in this example. The schema of the Account object will be automatically filled.
4. Click the [...] button next to Edit schema and in the pop-up dialog box, remove all schema columns except the three columns Id, Name, and talendlena__Account_External_ID__c. When done, click OK to save the changes and in the pop-up dialog box, click Yes to propagate the schema to the next tLogRow component.
5. Select Query from the Query Mode drop-down list and specify the condition used to filter data to be retrieved. In this example, it is Name like '%talend%'.

All Salesforce Account records whose Name field contains talend will be retrieved.
6. Double-click the third tLogRow component and on its Basic settings view, select Table in the Mode area to display the result in a table.

Retrieving updated Salesforce Contact records using SOQL query
Procedure

1. Double-click the third **tSalesforceInput** component to open its **Basic settings** view on the **Component** tab.

2. Select the component that will create the Salesforce connection from the **Connection Component** drop-down list, **tSalesforceConnection_1** in this example.

3. Click the [...] button next to the **Module Name** field and in the pop-up dialog box, select the object from which data will be retrieved, **Contact** in this example. The schema of the Contact object will be automatically filled.

4. Click the [...] button next to **Edit schema** and in the pop-up dialog box, remove all schema columns except the four columns **Name**, **Phone**, **Email**, and **talendlena__Contact_Talend_ID__c**.

5. Add another two columns to retrieve name and external ID of the linked Account object, **Contact_Account_Name** and **Contact_Account_talendlena__Account_External_ID__c** in this example.

   To retrieve data of a linked object column, it is necessary to define the name of the column in a particular manner in the schema editor. The correct syntax is **NameofCurrentObject_NameofLinkedObject_NameofColumnofInterest**. If this syntax is not respected, data from the linked object will not be returned. For more information about how to set schema for the relationship query, see How to set schema for the guess query feature of **tSalesforceInput** on page 3410.

6. Click **OK** to save the changes. In the pop-up dialog box, click **Yes** to propagate the schema to the next **tLogRow** component.

7. Select **Query** from the **Query Mode** drop-down list.

8. Select the **Manual Query** check box and click the **Guess query** button to generate the SOQL query based on the defined module name and schema columns. The generated SOQL query will look like below.

   ```sql
   SELECT Name, Phone, Email, talendlena__Contact_Talend_ID__c,
   Contact.Account.Name, Contact_Account_talendlena__Account_External_ID__c FROM Contact
   ```

9. In the generated SOQL query string, replace each underscore character after the object name in the column name **Contact_Account_talendlena__Account_External_ID__c** with a dot character. Then add the **WHERE** condition clause used to filter data to be retrieved at the end of the generated SOQL query. In this example, it is **talendlena__Contact_Talend_ID__c like**
'201801%' and all Salesforce Contact records whose `talendlena__Contact_Talend_ID__c` field begins with '201801' will be retrieved. The updated SOQL query will look like below.

```sql
SELECT Name, Phone, Email, talendlena__Contact_Talend_ID__c, Contact.Account.Name, Contact.Account.talendlena__Account_External_ID__c FROM Contact WHERE talendlena__Contact_Talend_ID__c like '201801%'
```

10. Double-click the fourth `tLogRow` component and on its Basic settings view, select Table in the Mode area to display the result in a table.

**Executing the Job to upsert Salesforce data based on external ID**

**Procedure**

1. Press Ctrl + S to save your Job.
2. Press F6 to execute your Job.

As shown above, two Account records `account_talend` and `account_talend_doc` are successfully inserted, two Contact records Beckham David and Taylor Swift are successfully inserted and updated based on the mapping relationship with the external IDs in the Account object, and another Contact record missing the value for the required field `LastName` is not inserted.

You can also go to the Salesforce website to double-check the Job execution result.
<table>
<thead>
<tr>
<th>Action</th>
<th>First Name</th>
<th>Last Name</th>
<th>Account Name</th>
<th>Phone</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>David</td>
<td>Beckham</td>
<td>account_talent</td>
<td>+86 13908000088</td>
<td><a href="mailto:beckham.david@talend.com">beckham.david@talend.com</a></td>
</tr>
<tr>
<td></td>
<td>Swift</td>
<td>Taylor</td>
<td>account_talent_doc</td>
<td>+86 13908000088</td>
<td><a href="mailto:taylor.swift@talend.com">taylor.swift@talend.com</a></td>
</tr>
</tbody>
</table>
Generates the file to be processed by the tSalesforceBulkExec component for bulk processing. The tSalesforceOutputBulk and tSalesforceBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used to feed the Salesforce database. These two steps are fused together in the tSalesforceOutputBulkExec component. The advantage of using two separate steps is that the data can be transformed before it is loaded into Salesforce.

**tSalesforceOutputBulk Standard properties**

These properties are used to configure tSalesforceOutputBulk running in the Standard Job framework. The Standard tSalesforceOutputBulk component belongs to the Business and the Cloud families. The component in this framework is generally available.

### Basic settings

| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. Click **Sync columns** to retrieve the schema from the previous component connected in the Job. |
| **Bulk File Path** | Specify the path to the file to be generated. |
| **Append** | Select this check box to append new data at the end of the file if it already exists, instead of overwriting the existing data. |
| **Ignore Null** | Select this check box to ignore NULL values. |
### Relationship mapping for upsert

Click the [+] button to add lines as needed and specify the external ID fields in the input flow, the lookup relationship fields in the upsert module, the lookup module as well as the external ID fields in the lookup module.

- **Column name of Talend Schema**: the name of the external ID field in the input flow. It refers to the fields in the schema of the preceding component. Such columns are intended to match against the external ID fields specified in the **External id name** column, which are the fields of the lookup module specified in the **Module name** column.

- **Lookup relationship field name**: the name of the lookup relationship field. It refers to the lookup relationship fields of the module specified in the **Module Name** field in the **Basic settings** view of the tSalesforceBulkExec component and is intended to establish relationship with the lookup module specified in the **Module name** column in this table. For how to define the lookup relationship fields and how to provide their correct names in the **Lookup relationship field name** field, go to the Salesforce website and launch the Salesforce Data Loader application for proper actions and information.

- **Module name**: the name of the lookup module.

- **Polymorphic**: select this check box when and only when polymorphic fields are used for relationship mapping. You will get an error if you keep this check box cleared for a polymorphic field or select it for a field that is not polymorphic. For more information about the polymorphic fields, search **polymorphic** at [http://www.salesforce.com/us/developer/docs/api_asynch/](http://www.salesforce.com/us/developer/docs/api_asynch/).

- **External id name**: the name of the external ID field in the lookup module specified in the **Module name** column.

### tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>
Usage

| Usage rule | This component is more commonly used with the tSalesforceBulkExec component. Used together, they gain performance while feeding or modifying information in Salesforce.com. |

Related scenario

No scenario is available for this component yet.
tSalesforceOutputBulkExec

Bulk-loads data in a given file into a Salesforce object.

The tSalesforceOutputBulk and tSalesforceBulkExec components are used together in a two step process. In the first step, an output file is generated. In the second step, this file is used to feed the Salesforce database. These two steps are fused together in the tSalesforceOutputBulkExec component. The advantage of using two separate steps is that the data can be transformed before it is loaded into Salesforce.

**tSalesforceOutputBulkExec Standard properties**

These properties are used to configure tSalesforceOutputBulkExec running in the Standard Job framework.

The Standard tSalesforceOutputBulkExec component belongs to the Business and the Cloud families. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select the way the connection details will be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

This property is not available when other connection component is selected from the Connection Component drop-down list.

<table>
<thead>
<tr>
<th>Connection Component</th>
<th>Select the component that opens the database connection to be reused by this component.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Connection type</th>
<th>Select the type of the connection from the drop-down list. The connection properties will vary according to the connection type you choose.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Basic</strong>: select this option to access Salesforce by entering your Salesforce username and password.</td>
</tr>
<tr>
<td></td>
<td>• <strong>OAuth</strong>: select this option to access Salesforce using OAuth (Open Authorization) 2.0 protocol to authenticate the user. From the OAuth2 flow type drop-down list displayed, you need to choose an OAuth 2.0 authentication flow type, either Json Web Token Flow or Implicit Flow (Deprecated) (User Agent Flow). This way, you need to create a connected app in Salesforce to configure a Salesforce authentication provider.</td>
</tr>
</tbody>
</table>
For more information about the OAuth authentication flow, see Authenticate Apps with OAuth.

For more information about how to create a connected app in Salesforce, see Create a Connected App.

<table>
<thead>
<tr>
<th><strong>User Id</strong></th>
<th>The Salesforce username. This property is available only when the <strong>Basic</strong> connection type is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Password</strong></td>
<td>The Salesforce password associated with the username. This property is available only when the <strong>Basic</strong> connection type is selected.</td>
</tr>
<tr>
<td><strong>Security Token</strong></td>
<td>The Salesforce security token. For more information, see Reset Your Security Token. This property is available only when the <strong>Basic</strong> connection type is selected.</td>
</tr>
<tr>
<td><strong>Issuer</strong></td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see Create a Connected App. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Subject</strong></td>
<td>The Salesforce username. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Expiration time (in seconds)</strong></td>
<td>The expiration time of the assertion (in seconds) within five minutes. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Key store</strong></td>
<td>The path to the keystore file in Java Keystore (JKS) format. The keystore file can be generated by creating a certificate signed by Salesforce and then exporting it to keystore. For more information, see Generate a Self-Signed Certificate. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Key store password</strong></td>
<td>The keystore password. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Certificate alias</strong></td>
<td>The unique name of the certificate signed by Salesforce. This property is available only when the OAuth <strong>Json Web Token Flow</strong> type is selected.</td>
</tr>
<tr>
<td><strong>Client Id</strong></td>
<td>The OAuth Consumer Key, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see Create a Connected App. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td><strong>Client Secret</strong></td>
<td>The OAuth Consumer Secret, generated when your connected app is created and shown in the connected app detail page in Salesforce. For more information, see Create a Connected App. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td><strong>Callback Host</strong></td>
<td>The host value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the API (Enable OAuth Settings) area of the connected app detail page in Salesforce. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td><strong>Callback Port</strong></td>
<td>The port value in the OAuth authentication callback URL that is defined during the creation of a connected app and will be shown in the API (Enable OAuth Settings) area of the connected app detail page in Salesforce. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td><strong>Token File</strong></td>
<td>The path to the token file that stores the refresh token used to get the access token without authorization. This property is available only when the OAuth Implicit Flow type is selected.</td>
</tr>
<tr>
<td><strong>Module Name</strong></td>
<td>Click the [...] button next to the field and in the dialog box displayed, select the module that will be used or select the Use custom object check box and specify the module name in the Object Name field.</td>
</tr>
</tbody>
</table>
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No. |
upon completion and choose this schema metadata again in the [Repository Content] window.

Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

### Output Action

Select one of the following operations to be performed from the drop-down list.

- **INSERT**: insert one or more new records into Salesforce.
- **UPDATE**: update one or more existing records in Salesforce.
- **UPSERT**: create new records and update existing records. In the **Upsert Key Column** field displayed, specify the key column for the upsert operation.
- **DELETE**: delete one or more records in Salesforce.

### Bulk File Path

Specify the path to the file that stores the data to be processed.

### Append

Select this check box to append new data at the end of the file if it already exists, instead of overwriting the existing data.

### Ignore Null

Select this check box to ignore NULL values.

### Advanced settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salesforce URL</td>
<td>The Webservice URL required to connect to Salesforce.</td>
</tr>
<tr>
<td>API version</td>
<td>The Salesforce API version. This property is available only when the OAuth connection type is selected.</td>
</tr>
<tr>
<td>Need compression</td>
<td>Select this check box to activate SOAP message compression, which can result in increased performance levels.</td>
</tr>
<tr>
<td>Trace HTTP message</td>
<td>Select this check box to output the HTTP interactions on the console.</td>
</tr>
<tr>
<td>Client Id</td>
<td>Enter the ID of the real user to differentiate between those who use the same account and password to access the Salesforce website.</td>
</tr>
<tr>
<td>Timeout</td>
<td>Enter the intended number of query timeout in milliseconds in Salesforce.</td>
</tr>
<tr>
<td>Use Proxy</td>
<td>Select this check box to use a proxy server, and in the Host, Port, User Id, and Password fields displayed, specify the connection parameters of the proxy server.</td>
</tr>
<tr>
<td>Relationship mapping for upsert</td>
<td>Click the [*] button to add lines as needed and specify the external ID fields in the input flow, the lookup relationship fields in the upsert module, the lookup</td>
</tr>
</tbody>
</table>
module as well as the external ID fields in the lookup module.

- **Column name of Talend Schema**: the name of the external ID field in the input flow. It refers to the fields in the schema of the preceding component. Such columns are intended to match against the external ID fields specified in the **External id name** column, which are the fields of the lookup module specified in the **Module name** column.

- **Lookup relationship field name**: the name of the lookup relationship field. It refers to the lookup relationship fields of the module specified in the **Module Name** field in the **Basic settings** view and is intended to establish relationship with the lookup module specified in the **Module name** column in this table. For how to define the lookup relationship fields and how to provide their correct names in the **Lookup relationship field name** field, go to the Salesforce website and launch the Salesforce Data Loader application for proper actions and information.

- **Module name**: the name of the lookup module.

- **Polymorphic**: select this check box when and only when polymorphic fields are used for relationship mapping. You will get an error if you keep this check box cleared for a polymorphic field or select it for a field that is not polymorphic. For more information about the polymorphic fields, search **polymorphic** at http://www.salesforce.com/us/developer/docs/api_asynch/.

- **External id name**: the name of the external ID field in the lookup module specified in the **Module name** column.

This property is available only when **UPSERT** is selected from the **Output Action** drop-down list.

<table>
<thead>
<tr>
<th>Concurrency Mode</th>
<th>Select the concurrency mode for the job.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Parallel</strong>: process batches in parallel mode.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Serial</strong>: process batches in serial mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rows to Commit</th>
<th>Specify the number of lines per data batch to be processed.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bytes to Commit</th>
<th>Specify the number of bytes per data batch to be processed.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Wait Time Check Batch State</th>
<th>Specify the wait time (in milliseconds) for checking whether the batches in a Job have been processed until all batches are finally processed.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>
Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage rule</td>
<td>This component is mainly used when no particular transformation is required on the data to be loaded into Salesforce.</td>
</tr>
<tr>
<td>Limitation</td>
<td>The bulk data to be processed in Salesforce should be in .csv format.</td>
</tr>
</tbody>
</table>

Inserting bulk data into Salesforce

This scenario describes a four-component Job that bulk-loads data in a file into Salesforce, performs the intended action on the data, and ends up with displaying the Job execution results on the console.

The content of the input file `SalesforceAccount.txt` used in this example is as follows:

Name;ParentId;Phone;Fax  
Burlington Textiles Corp of America;;(336) 222-7000;(336) 222-8000  
Dickenson plc;; (785) 241-6200;(785) 241-6201  
GenePoint;;(650) 867-3450;(650) 867-9895  
Edge Communications;talend;(512) 757-6000;(512) 757-9000  
Grand Hotels & Resorts Ltd;talend;(312) 596-1000;(312) 596-1500

Setting up the Job for inserting bulk data into Salesforce

Procedure

1. Create a new Job and add a `tFileInputDelimited` component, a `tSalesforceOutputBulkExec` component, and two `tLogRow` components by typing their names on the design workspace or dropping them from the Palette.
2. Link the `tFileInputDelimited` component to the `tSalesforceOutputBulkExec` component using a Row > Main connection.
3. Link the `tSalesforceOutputBulkExec` component to the first `tLogRow` component using a Row > Main connection.
4. Link the tSalesforceOutputBulkExec component to the second tLogRow component using a Row > Reject connection.

Configuring the Job for inserting bulk data into Salesforce

Procedure

1. Double-click the tFileInputDelimited component to open its Basic settings view.

   ![FileInputDelimited Basic settings](image1)

   2. In the File name/Stream field, browse to or enter the path to the input data file. In this example, it is D:/SalesforceAccount.txt.

   3. Click the [...] button next to Edit schema and in the pop-up schema dialog box, define the schema by adding four columns Name, ParentId, Phone and Fax of String type. When done, click OK to save the changes and close the dialog box.

   ![Schema of tFileInputDelimited](image2)

4. Double-click the tSalesforceOutputBulkExec component to open its Basic settings view.
5. In the **User Id**, **Password** and **Security Key** fields, enter the user authentication information required to access Salesforce.

6. Click the [...] button next to the **Module Name** field and in the pop-up dialog box, select the object you want to access. In this example, it is *Account*.

7. In the **Bulk File Path** field, browse to or enter the path to the CSV file that stores the data for bulk processing. The bulk file to be processed must be in csv format.

8. Double-click the first `tLogRow` component to open its **Basic settings** view.

9. In the **Mode** area, select **Table (print values in cells of a table)** for better readability of the results.

10. Do the same to configure the second `tLogRow` component.

**Executing the Job to insert bulk data into Salesforce**

**Procedure**

1. Press **Ctrl + S** to save the Job.

2. Press **F6** to execute the Job.

   On the console of the **Run** view, you can check the execution result.
In the tLogRow_1 table, you can read the data inserted into Salesforce.

In the tLogRow_2 table, you can read the rejected data due to the incompatibility with the Account objects you have accessed.

Note that if you want to transform the input data before loading them into Salesforce, you need to use `tSalesforceOutputBulk` and `tSalesforceBulkExec` in cooperation to achieve this purpose.
tSalesforceWaveBulkExec

Loads data into Salesforce Analytics Cloud from a local file.

**tSalesforceWaveBulkExec Standard properties**

These properties are used to configure tSalesforceWaveBulkExec running in the Standard Job framework.

The Standard tSalesforceWaveBulkExec component belongs to the Business and the Cloud families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

Note that when a Job contains the parent Job and the child Job, Component List presents only the connection components in the same Job level.

**User Name and Password**

Enter the Web service authentication details.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**End Point**

Enter the WebService URL required to connect to Salesforce. For example, https://login.salesforce.com/services/Soap/u/37.0. Note that the version in the URL should be 32.0 or later.

**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

- For the fields of numeric type (for example, byte, short, int, long, float, double, and BigDecimal), their length and precision values need to be specified. The default length and precision values are 10 and 2, and you can also specify their custom values in the schema editor.
- For the fields of date type, you need to specify the format of the date in the schema editor. For more
information about the supported date format, see Analytics Cloud External Data Format Reference.

The Schema list and the Editor schema button disappear if the Custom JSON Metadata check box is selected.

**Built-In:** You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

**Repository:** You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to Built-in for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Operation**

Select an operation to perform on the dataset:

- **Append:** Append all data to the dataset. Create a dataset if it does not exist.
- **Upsert:** Insert or update rows in the dataset. Create a dataset if it does not exist.
- **Overwrite:** Create a new dataset with the given data, and replace dataset if it already exists.
- **Delete:** Delete the rows from the dataset.

**Note:**

- A metadata JSON file is required for the Append, Upsert, and Delete operations.
- The data and metadata for the Append and Upsert operations must match the dataset on which the operation is happening. All columns, dimensions, and measures must match exactly.
- The Append operation is not allowed if you specify any column as the primary key.
- You must specify one (and only one) column as the primary key on which the Upsert or Delete operation is based. You can do that by clicking Edit schema and
selecting the check box next to the column you want to set as the primary key.

- The metadata for the **Delete** operation must be a subset of the dataset columns.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type in the name of the dataset into which the data will be loaded.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV File</td>
<td>Specify the path to the local CSV file to be loaded.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| CSV Encoding | Enter the encoding type of the CSV file.  
This field is not visible when the **Custom JSON Metadata** check box is selected. |
|--------------|-------------------------------------------------------------------|
| Fields Delimiter | Enter the character that separates the field values in the CSV file.  
This field is not visible when the **Custom JSON Metadata** check box is selected. |
| Fields Enclosed By | Enter the character used to enclose the field values in the CSV file.  
This field is not visible when the **Custom JSON Metadata** check box is selected. |
| Line Terminated By | Enter the character indicating the end of a line.  
This field is not visible when the **Custom JSON Metadata** check box is selected. |
| Auto Generate JSON Metadata Description | Select this check box to generate the JSON metadata description automatically. |
| Header | Specify the number of lines to ignore in the CSV file.  
This field is available only when the **Auto Generate JSON Metadata Description** check box is cleared. |
| Unique API Name | Specify the unique API name for the object in the JSON metadata description.  
This field is available only when the **Auto Generate JSON Metadata Description** check box is cleared. |
| Label | Specify the display name for the object in the JSON metadata description.  
This field is available only when the **Auto Generate JSON Metadata Description** check box is cleared. |
| Fully Qualified Name | Specify the full path that uniquely identifies the record in the JSON metadata description.  
This field is available only when the **Auto Generate JSON Metadata Description** check box is cleared. |
<table>
<thead>
<tr>
<th><strong>Custom JSON Metadata</strong></th>
<th>Select this check box to use a customized JSON metadata file. This check box is available only when the <strong>Auto Generate JSON Metadata Description</strong> check box is cleared.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JSON Metadata</strong></td>
<td>Specify the path to the customized JSON metadata file. This field is available only when the <strong>Custom JSON Metadata</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Generate JSON in File</strong></td>
<td>Select this check box to write the JSON metadata description into a local file. This check box is not visible when the <strong>Custom JSON Metadata</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Generated JSON Folder</strong></td>
<td>Specify the directory where you want to store the generated JSON metadata file. This field is available only when the <strong>Generate JSON in File</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Retrieve Upload Status</strong></td>
<td>Select this check box to retrieve the status of the data upload.</td>
</tr>
<tr>
<td><strong>Time to wait for server answer (seconds)</strong></td>
<td>Specify the amount of time in seconds to wait for the upload status response from the server. This field is available only when the <strong>Retrieve Upload Status</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

| **ERROR_MESSAGE** | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

**Usage**

| **Usage rule** | This component can be used as a standalone component. |

**Related scenario**

No scenario is available for this component yet.
tSalesforceWaveOutputBulkExec

Gains in performance during data operations to the Salesforce Analytics Cloud.

tSalesforceWaveOutputBulkExec receives data from the preceding component, generates a local CSV file and then loads data from the file to the Salesforce Analytics Cloud.

**tSalesforceWaveOutputBulkExec Standard properties**

These properties are used to configure tSalesforceWaveOutputBulkExec running in the Standard Job framework.

The Standard tSalesforceWaveOutputBulkExec component belongs to the Business and the Cloud families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either <strong>Built-In</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The connection fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined. Note that when a Job contains the parent Job and the child Job, <strong>Component List</strong> presents only the connection components in the same Job level.</td>
</tr>
<tr>
<td><strong>User Name</strong> and <strong>Password</strong></td>
<td>Enter the Web service authentication details. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>End Point</strong></td>
<td>Enter the WebService URL required to connect to Salesforce. For example, <a href="https://login.salesforce.com/services/Soap/u/37.0">https://login.salesforce.com/services/Soap/u/37.0</a>. Note that the version in the URL should be 32.0 or later.</td>
</tr>
</tbody>
</table>
| **Schema** and **Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
  • For the fields of numeric type (for example, byte, short, int, long, float, double, and BigDecimal), their length and precision values need to be specified. The default length and precision values are 10 and 2, |
and you can also specify their custom values in the schema editor.

- For the fields of date type, you need to specify the format of the date in the schema editor. For more information about the supported date format, see Analytics Cloud External Data Format Reference.

The Schema list and the Editor schema button disappear if the Custom JSON Metadata check box is selected.

**Built-In:** You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

**Repository:** You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to Built-in for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

Click Sync columns to retrieve the schema from the previous component connected in the Job.

**Operation**

Select an operation to perform on the dataset:

- **Append:** Append all data to the dataset. Create a dataset if it does not exist.
- **Upsert:** Insert or update rows in the dataset. Create a dataset if it does not exist.
- **Overwrite:** Create a new dataset with the given data, and replace dataset if it already exists.
- **Delete:** Delete the rows from the dataset.

**Note:**

- A metadata JSON file is required for the Append, Upsert, and Delete operations.
- The data and metadata for the Append and Upsert operations must match the dataset on which the operation is happening. All columns, dimensions, and measures must match exactly.
The Append operation is not allowed if you specify any column as the primary key.

- You must specify one (and only one) column as the primary key on which the Upsert or Delete operation is based. You can do that by clicking Edit schema and selecting the check box next to the column you want to set as the primary key.
- The metadata for the Delete operation must be a subset of the dataset columns.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type in the name of the dataset into which the data will be loaded.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generated Temp CSV File</td>
<td>Specify the local path to the CSV file to be generated.</td>
</tr>
<tr>
<td>Append</td>
<td>Select this check box to append data to the specified local file if it already exists, instead of overwriting it.</td>
</tr>
<tr>
<td>Include Header</td>
<td>Select this check box to include the column header to the file.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>CSV Encoding</th>
<th>Enter the encoding type of the CSV file. Note that the value of this field should be same as the value defined in the JSON metadata file when the Custom JSON Metadata check box is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields Delimiter</td>
<td>Enter the character that separates the field values in the CSV file. Note that the value of this field should be same as the value defined in the JSON metadata file when the Custom JSON Metadata check box is selected.</td>
</tr>
<tr>
<td>Fields Enclosed By</td>
<td>Enter the character used to enclose the field values in the CSV file. Note that the value of this field should be same as the value defined in the JSON metadata file when the Custom JSON Metadata check box is selected.</td>
</tr>
<tr>
<td>Line Terminated By</td>
<td>Enter the character indicating the end of a line. Note that the value of this field should be same as the value defined in the JSON metadata file when the Custom JSON Metadata check box is selected.</td>
</tr>
<tr>
<td>Create directory if not exists</td>
<td>Select this check box to create the directory specified in the Generated Temp CSV File field if it does not exist.</td>
</tr>
<tr>
<td>Custom the flush buffer size</td>
<td>Select this check box to specify the number of lines to write before emptying the buffer.</td>
</tr>
<tr>
<td>Row number</td>
<td>Specify the number of lines to write before emptying the buffer.</td>
</tr>
<tr>
<td><strong>Field</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Don't generate empty file</td>
<td>Select this check box if you do not want to generate empty files.</td>
</tr>
<tr>
<td>Auto Generate JSON Metadata Description</td>
<td>Select this check box to generate the JSON metadata description automatically.</td>
</tr>
<tr>
<td>Header</td>
<td>Specify the number of lines to ignore in the CSV file.</td>
</tr>
<tr>
<td>Unique API Name</td>
<td>Specify the unique API name for the object in the JSON metadata description.</td>
</tr>
<tr>
<td>Label</td>
<td>Specify the display name for the object in the JSON metadata description.</td>
</tr>
<tr>
<td>Fully Qualified Name</td>
<td>Specify the full path that uniquely identifies the record in the JSON metadata description.</td>
</tr>
<tr>
<td>Custom JSON Metadata</td>
<td>Select this check box to use a customized JSON metadata file.</td>
</tr>
<tr>
<td>JSON Metadata</td>
<td>Specify the path to the customized JSON metadata file.</td>
</tr>
<tr>
<td>Generate JSON in File</td>
<td>Select this check box to write the JSON metadata description into a local file.</td>
</tr>
<tr>
<td>Generated JSON Folder</td>
<td>Specify the directory where you want to store the generated JSON metadata file.</td>
</tr>
<tr>
<td>Retrieve Upload Status</td>
<td>Select this check box to retrieve the status of the data upload.</td>
</tr>
<tr>
<td>Time to wait for server answer (seconds)</td>
<td>Specify the amount of time in seconds to wait for the upload status response from the server.</td>
</tr>
</tbody>
</table>

*This field is available only when the Custom the flush buffer size check box is selected.*

*This field is available only when the Auto Generate JSON Metadata Description check box is cleared.*

*This field is available only when the Auto Generate JSON Metadata Description check box is cleared.*

*This field is available only when the Custom JSON Metadata check box is selected.*
### tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is mainly used when no particular transformation is required on the data to be loaded to Salesforce Analytics Cloud.</th>
</tr>
</thead>
</table>

### Related scenario

For a similar scenario, see Inserting bulk data into Salesforce on page 3439.
tSampleRow

Selects rows according to a list of single lines and/or a list of groups of lines.
tSampleRow filters rows according to line numbers.

**tSampleRow Standard properties**

These properties are used to configure tSampleRow running in the Standard Job framework.
The Standard tSampleRow component belongs to the Processing family.
The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
- View schema: choose this option to view the schema only.
- Change to built-in property: choose this option to change the schema to Built-in for local changes.
- Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. Click Sync columns to retrieve the schema from the previous component in the Job. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>Enter a range using the relevant syntax to choose a list of single lines and/or a list of groups of lines.</td>
</tr>
</tbody>
</table>

**Global Variables**

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After |
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule | This component handles flows of data therefore it requires input and output components. |

## Scenario: Filtering rows and groups of rows

This scenario describes a three-component Job. A **tRowGenerator** is used to create random entries which are directly sent to a **tSampleRow** where they will be filtered according to a defined range. In this scenario, we suppose the input flow contains names of salespersons along with their respective number of sold products and their years of presence in the enterprise. The result of the filtering operation is displayed on the **Run** console.

### Dropping and linking the components

**Procedure**

1. Drop the following components from the **Palette** onto the design workspace: **tRowGenerator**, **tSampleRow**, and **tLogRow**.
2. Connect the three components using **Row > Main** links.

### Results

![Diagram of components](image)

### Configuring the components

**Procedure**

1. In the design workspace, select **tRowgenerator**, and click the **Component** tab to define the basic settings for **tRowGenerator**.
2. Click the [...] button next to **Edit Schema** to define the data you want to use as input. In this scenario, the schema is made of five columns.
3. In the **Basic settings** view, click **RowGenerator Editor** to define the data to be generated.

4. In the **RowGenerator Editor**, specify the number of rows to be generated in the **Number of Rows for RowGenerator** field and click **OK**. The **RowGenerator Editor** closes.

5. In the design workspace, select **tSampleRow** and click the **Component** tab to define the basic settings for **tSampleRow**.

6. In the **Basic settings** view, set the **Schema** to **Built-In** and click **Sync columns** to retrieve the schema from the **tRowGenerator** component.

7. In the **Range** panel, set the filter to select your rows using the correct syntax as explained. In this scenario, we want to select the first and fifth lines along with the group of lines between 9 and 12.

8. In the design workspace, select **tLogRow** and click the **Component** tab to define its basic settings. For more information about **tLogRow**, see **tLogRow** on page 1988.
Saving and executing the Job

Procedure

1. Press **Ctrl+S** to save your Job.
2. Press **F6**, or click **Run** on the **Run** tab to execute the Job.

Results

```
Starting job sample_Row at 11:16 20/08/2008.
+---------------+-----------+------------+------------+--------------+
| ID  | Years_in_Enterprise | First_Name | Last_Name  | Sales_Value  |
+-----+---------------------+------------+------------+--------------+
| 1   | 1                   | Martin     | Taft       | 16           |
| 5   | 1                   | Franklin   | Adams      | 45           |
| 9   | 1                   | Villian    | Madison    | 15           |
| 10  | 1                   | George     | Kennedy    | 47           |
| 11  | 1                   | George     | Van Buren  | 16           |
| 12  | 1                   | Theodore   | Grant      | 68           |
+-----+---------------------+------------+------------+--------------+
Job sample_Row ended at 11:16 20/08/2008. [exit code=0]
```

The filtering result displayed on the console shows the first and fifth rows and the group of rows between 9 and 12.
tSAPHanaClose

Closes a connection to a SAP Hana database.

**tSAPHanaClose Standard properties**

These properties are used to configure tSAPHanaClose running in the Standard Job framework.

The Standard tSAPHanaClose component belongs to the Databases family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tSAPHanaConnection component in the list if more than one connection is planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is to be used along with SAP Hana components, especially with tSAPHanaConnection and tSAPHanaCommit. |

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tSAPHanaCommit

Commits in one go, using a unique connection, a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

tSAPHanaCommit validates the data processed through the Job into the connected database.

**tSAPHanaCommit Standard properties**

These properties are used to configure tSAPHanaCommit running in the Standard Job framework.

The Standard tSAPHanaCommit component belongs to the Databases family.

The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tSAPHanaConnection component in the list if more than one connection is planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Warning:**
If you want to use a Row > Main connection to link tSAPHanaCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tSAPHana* components, especially with the |
tSAPHanaCommit

| tSAPHanaConnection and tSAPHanaRollback components. Use this component if the Auto Commit option of the tSAPHanaConnection component is cleared. |

### Related scenario

For a related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476.
tSAPHanaConnection

Establishes a SAP Hana connection to be reused by other SAP Hana components in your Job. tSAPHanaConnection opens a connection to the database for a current transaction.

tSAPHanaConnection Standard properties

These properties are used to configure tSAPHanaConnection running in the Standard Job framework. The Standard tSAPHanaConnection component belongs to the Databases and the ELT families. The component in this framework is available when you are using one of the Talend solutions with Big Data.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the SAP Hana Database (HDB) version you are using.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of database server.</td>
</tr>
<tr>
<td>Table Schema</td>
<td>Name of the database schema.</td>
</tr>
</tbody>
</table>
| Username and Password | Database authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Additional JDBC Parameters | Specify additional connection properties in the database connection you are creating. |

Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
</table>
| Auto Commit    | Select this check box to commit any changes to the database automatically upon the transaction.  
With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and |
changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Usage**

| Usage rule | This component is more commonly used with other tSAP* components, especially with the tSAPHanaCommit and tSAPHanaRollback components. |

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tSAPHanaInput

Executes a database query with a defined command which must correspond to the schema definition. tSAPHanaInput reads a database and extracts fields based on a query. Then it passes on rows to the next component via a Main row link.

**tSAPHanaInput Standard properties**

These properties are used to configure tSAPHanaInput running in the Standard Job framework. The Standard tSAPHanaInput component belongs to the Databases family.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the SAP Hana Database (HDB) version you are using.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of database server.</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Name of the database schema.</td>
</tr>
</tbody>
</table>
| **Username and Password** | Database authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. |
| | **Built-in**: The schema is created and stored locally for this component only. For more information, see Talend Studio User Guide. |
| | **Repository**: The schema already exists and is stored in the Repository, hence can be reused. For more information, see Talend Studio User Guide. |
| | Click **Edit schema** to make changes to the schema. If the current schema is of the Repository type, three options are available:  
• **View schema**: choose this option to view the schema only.  
• **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
| **Table Name** | Name of the table to be written. Note that only one table can be written at a time. |
| **Query Type** | Either Built-in or Repository. |
| | **Built-in**: Fill in manually the query statement or build it graphically using SQLBuilder. |
| | **Repository**: Select the relevant query stored in the Repository. The Query field gets accordingly filled in. |
| **Guess Query** | Click the Guess Query button to generate the query which corresponds to your table schema in the Query field. |
| **Guess schema** | Click the Guess schema button to retrieve the table schema. |
Query

Enter your database query paying particularly attention to properly sequence the fields in order to match the schema definition.

Advanced settings

<table>
<thead>
<tr>
<th><strong>Additional JDBC Parameters</strong></th>
<th>Specify additional connection properties in the database connection you are creating. This option is not available if you have selected Use an existing connection check box in the Basic settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trim all the String/Char columns</strong></td>
<td>Select this check box to remove leading and trailing whitespaces from all the String/Char columns.</td>
</tr>
</tbody>
</table>
| **Trim column** | Remove leading and trailing whitespaces from defined columns.  

**Note:**  
Clear Trim all the String/Char columns check box to enable this option. |
| **tStatCatcher Statistics** | Select this check box to collect log data at the component level. |

Global Variables

| **Global Variables** | **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

Usage

| **Usage rule** | This component is standalone as it includes the SAP Hana engine. This is a startable component that can initiate a data flow processing. |
| **Dynamic settings** | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple |
connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

No scenario is available for the Standard version of this component yet.
tSAPHanaOutput

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tSAPHanaOutput writes, updates, makes changes or suppresses entries in a SAP Hana database.

**tSAPHanaOutput Standard properties**

These properties are used to configure tSAPHanaOutput running in the Standard Job framework.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined. <strong>Note:</strong> When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to: 1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection. 2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see <a href="#">Talend Studio User Guide</a>.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the SAP Hana Database (HDB) version you are using.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of database server.</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td><strong>Table Schema</strong></td>
<td>Name of the database schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Database authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
</tbody>
</table>
| **Action on table** | On the table defined, you can perform one of the following operations:  
  - **None**: No operation is carried out.  
  - **Drop and create table**: The table is removed and created again.  
  - **Create table**: The table does not exist and gets created.  
  - **Create table if not exists**: The table is created if it does not exist.  
  - **Drop table if exists and create**: The table is removed if it already exists and created again.  
  - **Clear table**: The table content is deleted. |
| **Type of table storage** | This option is only available if you create (with or without drop) the database table. This option allows you to define the way the data is stored in the table. The following types of table storage organization are available:  
  - **Row**: Data is stored in rows. It is preferable to use this table type if the majority of table access involves selecting a few records, with all attributes selected.  
  - **Column**: Data is stored in columns. It is preferable to use this table type if the majority of table access will be through a large number of tuples, with only a few selected attributes.  
  - **History column**: Creates a table with a particular transaction session type called HISTORY. Tables with session type HISTORY support time travel queries. For more information on HISTORY and time travel queries, see [http://help.sap.com/saphelp_hanaplatform/helpdata/en/20/d56cd075191014b7ea9e4e77477aae/content.htm](http://help.sap.com/saphelp_hanaplatform/helpdata/en/20/d56cd075191014b7ea9e4e77477aae/content.htm).  
  - **Global temporary**: The table definition and data can be seen by any user and any connection during the current session. The table is truncated at the end of the session.  
  - **Local temporary**: The table definition and data can only be seen by the user who created the table. |
| Action on data | On the data of the table defined, you can perform: 
**Insert**: Add new entries to the table. If duplicates are found, the Job stops. 
**Update**: Make changes to existing entries. 
**Insert or update**: Insert a new record. If the record with the given reference already exists, an update would be made. 
**Update or insert**: Update the record with the given reference. If the record does not exist, a new record would be inserted. 
**Delete**: Remove entries corresponding to the input flow. 

Note that it is necessary to specify at least one column as a primary key on which the **Update** and **Delete** operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, see **Use filed options** in the **Advanced settings** view. |

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields. 

**Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**. 

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**. 

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. 

You can find more details about how to verify default values in retrieved schema in Talend Help Center [https://help.talend.com](https://help.talend.com). |

|  | Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available: |
|  | • **View schema**: choose this option to view the schema only. |
|  | • **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes. |
|  | • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate |
the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th><strong>Die on error</strong></th>
<th>This check box is cleared by default. This means that <strong>Die on error</strong> skips the row when an error is encountered and completes the process for rows without errors.</th>
</tr>
</thead>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Additional JDBC Parameters</strong></th>
<th>Specify additional connection properties in the database connection you are creating. This option is not available if you have selected <strong>Use an existing connection</strong> check box in the Basic settings.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Commit every</strong></th>
<th>Enter the number of rows to be completed before committing batches of rows together into the database. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Additional Columns</strong></th>
<th>This option is not available if you create (with or without drop) the database table. This option allows you to call SQL functions to perform actions on columns, which are not <strong>insert</strong>, nor <strong>update</strong> or <strong>delete</strong> actions, or any other action that requires particular preprocessing.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th>Type in the name of the schema column to be altered or inserted as new column</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>SQL expression</strong></th>
<th>Type in the SQL statement to be executed in order to alter or insert the relevant column data.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Position</strong></th>
<th>Select <strong>Before</strong>, <strong>Replace</strong> or <strong>After</strong> following the action to be performed on the reference column.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Reference column</strong></th>
<th>Type in a column of reference that tSAPHanaOutput can use to locate or replace the new column or the column to be modified.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Use field options</strong></th>
<th>Select this check box to customize a request, especially when there is double action on data. In that table, you can define: <strong>Key in update</strong>: define the primary key to use in an update operation. <strong>Key in delete</strong>: define the primary key to use in a delete operation. <strong>Updatable</strong>: define the column(s) to update. <strong>Insertable</strong>: define the column(s) to insert.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Enable debug mode</strong></th>
<th>Select this check box to display each step during processing entries in a database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support null in “SQL WHERE” statement</td>
<td>Select this check box to validate null in “SQL WHERE” statement.</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>Use Batch Size</td>
<td>When selected, enables you to define the number of lines in each processed batch. This option is available only when you do not Use an existing connection in Basic settings.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_LINE_UPDATED: the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_INSERTED: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_DELETED: the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE_REJECTED: the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>QUERY: the query statement processed. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component must be connected to an Input component. It allows you to carry out actions on a table or on the data of a table in an SAP Hana database. It also allows you to create reject flows using a Row &gt; Reject link to filter erroneous data.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having</td>
</tr>
</tbody>
</table>
the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

---

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tSAPHanaRollback

Avoids to commit part of a transaction involuntarily.

`tSAPHanaRollback` cancels the transaction in the connected database.

`tSAPHanaRollback Standard properties

These properties are used to configure `tSAPHanaRollback` running in the Standard Job framework.

The Standard `tSAPHanaRollback` component belongs to the Databases family.

The component in this framework is available when you are using one of the `Talend` solutions with Big Data.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <code>tSAPHanaConnection</code> component in the list if more than one connection is planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other `tSAPHana*` components, especially with the `tSAPHanaConnection` and `tSAPHanaCommit` components. |

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tSAPHanaRow

Acts on the actual database structure or on the data (although without handling data).

tSAPHanaRow is the specific component for this database query. It executes the SQL query stated onto the specified database. The row suffix means the component implements a flow in the Job design although it does not provide output.

The SQLBuilder tool helps you write easily your SQL statements.

**tSAPHanaRow Standard properties**

These properties are used to configure tSAPHanaRow running in the Standard Job framework.

The Standard tSAPHanaRow component belongs to the Databases family.

The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see *Dynamic database components* on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
</table>
| Use an existing connection | Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<p>| DB Version | Select the SAP Hana Database (HDB) version you are using. |</p>
<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>:</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong>:</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of database server.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Database authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-in</strong>:</td>
<td>The schema is created and stored locally for this component only. For more information, see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Repository</strong>:</td>
<td>The schema already exists and is stored in the Repository, hence can be reused. For more information, see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Table Name</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
<tr>
<td><strong>Query Type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>:</td>
<td>Fill in manually the query statement or build it graphically using SQLBuilder.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</td>
</tr>
<tr>
<td><strong>Guess Query</strong></td>
<td>Click the Guess Query button to generate the query which corresponds to your table schema in the Query field.</td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>Enter your database query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is cleared by default. This means that Die on error skips the row when an error is encountered and completes the process for rows without errors.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **Additional JDBC Parameters** | Specify additional connection properties in the database connection you are creating. This option is not available if you have selected Use an existing connection check box in the Basic settings. |
| **Propagate QUERY’s recordset** | Select this check box to insert the result of the query into a column of the current flow. Select this column from the use column list. |

**Note:**
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of Object and this component is usually followed by tParseRecordSet. |

| **Commit every** | Number of rows to be completed before committing batches of rows together into the database. This option ensures transaction quality (but not rollback) and above all better performance on executions. |
| **Use PreparedStatement** | Select this check box if you want to query the database using a PreparedStatement. In the Set PreparedStatement Parameter table, define the parameters represented by “?” in the SQL instruction of the Query field in the Basic Settings tab. |

**Parameter Index:** Enter the parameter position in the SQL instruction. |

**Parameter Type:** Enter the parameter type. |

**Parameter Value:** Enter the parameter value. |

**Note:**
This option is very useful if you need to execute the same query several times. Performance levels are increased.
**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

---

### Global Variables

**Global Variables**

- **QUERY**: the query statement being processed. This is a Flow variable and it returns a string.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

---

### Usage

**Usage rule**

This component offers the flexibility of the database query and covers all possible SQL queries.

---

### Related scenarios

No scenario is available for the Standard version of this component yet.
tSasInput

Executes a DB query with a strictly defined statement which must correspond to the schema definition.

⚠️ **Warning:**

*Before being able to benefit from all functional objectives of the SAS components, make sure to install the following three modules: sas.core.jar, sas.intrnet.javatools.jar and sas.svc.connection.jar in the path lib > java in your Talend Studio directory. You can later verify, if needed whether the modules are successfully installed through the Modules view of the Studio.*

tSasInput reads a database and extracts fields based on a query. Then it passes on the field list to the component that follows via a **Row > Main** connection.

**tSasInput Standard properties**

These properties are used to configure tSasInput running in the Standard Job framework.

The Standard tSasInput component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

![icon]

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see *Talend Studio User Guide*.

**Host name**

SAS server IP address.

**Port**

Listening port number of server.

**Librefs**

Enter the directory name that holds the table to read followed by its access path. For example: `C:/SAS/TpSas`.

**Username** and **Password**

DB user authentication data.

To enter the password, click the `(...)` button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.
## Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

### Built-in

The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

### Repository

The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

## Table Name

Enter the name of the table to read preceded by the directory name that holds it. For example: "TpSas.Customers".

## Query type

The query can be **Built-in** for a particular job, or for commonly used query, it can be stored in the **Repository** to ease the query reuse.

## Query

If your query is not stored in the Repository, type in your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

## Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the job processing metadata at a Job level as well as at each component level.

## Global Variables

**Global Variables**

- **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.
- **QUERY**: the query statement being processed. This is a Flow variable and it returns a string.
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible SQL queries for databases using SAS connections.</th>
</tr>
</thead>
</table>

### Limitation

- Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### Related scenarios

For related topics, see:

- **Scenario**: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
tSasOutput

Executes the action defined on the table and/or on the data contained in the table, based on the incoming flow from the preceding component in the Job.

⚠️ Warning:

Before being able to benefit from all functional objectives of the SAS components, make sure to install the following three modules: sas.core.jar, sas.intrnet.javatools.jar and sas.svc.connection.jar in the path lib > java in your Talend Studio directory. You can later verify, if needed whether the modules are successfully installed through the Modules view of the Studio.

tSasOutput writes, updates, makes changes or suppresses entries in a database.

**tSasOutput Standard properties**

These properties are used to configure tSasOutput running in the Standard Job framework.
The Standard tSasOutput component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Use an existing connection</strong></th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td></td>
<td>For an example about how to share a database connection across Job levels, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

<p>| <strong>SAS URL</strong> | Enter the URL to connect to the desired DB. |
| <strong>Driver JAR</strong> | In the drop down list, select a desired available driver, or download one from a local directory through clicking the three-dot button. |
| <strong>Class Name</strong> | Type in the Class name to be pointed to in the driver. |</p>
<table>
<thead>
<tr>
<th><strong>Username and Password</strong></th>
<th>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to read.</td>
</tr>
<tr>
<td><strong>Action on data</strong></td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Make changes to existing entries</td>
</tr>
<tr>
<td></td>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td><strong>Update or insert</strong>: Update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Remove entries corresponding to the input flow.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning</strong>: It is necessary to specify at least one column as a primary key on which the <strong>Update</strong> and <strong>Delete</strong> operations are based. You can do that by clicking <strong>Edit Schema</strong> and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the <strong>Advanced settings</strong> view where you can simultaneously define primary keys for the <strong>Update</strong> and <strong>Delete</strong> operations. To do that: Select the <strong>Use field options</strong> check box and then in the <strong>Key in update column</strong>, select the check boxes next to the column names you want to use as a base for the <strong>Update</strong> operation. Do the same in the <strong>Key in delete column</strong> for the <strong>Delete</strong> operation.</td>
</tr>
<tr>
<td><strong>Clear data in table</strong></td>
<td>Select this check box to delete data in the selected table before any operation.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>
| | When the schema to be reused has default values that are integers or functions, ensure that these default }
values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Die on error**

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

### Advanced settings

**Commit every**

Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.

**Additional Columns**

This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.

- **Name**: Type in the name of the schema column to be altered or inserted as a new column.
- **SQL expression**: Type in the SQL statement to be executed in order to alter or insert the relevant column data.
- **Position**: Select **Before**, **Replace** or **After** following the action to be performed on the reference column.
- **Reference column**: Type in a column of reference that the **tSasOutput** can use to place or replace the new or altered column.

**Use field options**

Select this check box to customize a request, especially when there is double action on data.
Enable debug mode | Select this check box to display each step during processing entries in a database.

| tStatCatcher Statistics | Select this check box to collect log data at the component level.

Global Variables

| Global Variables | NB_LINE: the number of rows processed. This is an After variable and it returns an integer.
|                  | NB_LINE_UPDATED: the number of rows updated. This is an After variable and it returns an integer.
|                  | NB_LINE_INSERTED: the number of rows inserted. This is an After variable and it returns an integer.
|                  | NB_LINE_DELETED: the number of rows deleted. This is an After variable and it returns an integer.
|                  | NB_LINE_REJECTED: the number of rows rejected. This is an After variable and it returns an integer.
|                  | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

| Usage rule | This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.
|           | This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a SAS database. It also allows you to create a reject flow using a Row > Rejects link to filter data in error. For an example of tMySQLOutput in use, see Scenario: Retrieving data in error with a Reject link on page 2529.

Limitation

| Limitation | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find
Related scenarios

For scenarios in which `tSasOutput` might be used, see:

- **Scenario: Inserting a column and altering data using `tMysqlOutput`** on page 2521.
tSchemaComplianceCheck

Ensures the data quality of any source data against a reference data source.

tSchemaComplianceCheck validates all input rows against a reference schema or check types, nullability, length of rows against reference values. The validation can be carried out in full or partly.
# tSCPClose

Closes a connection to an SCP protocol. 
tSCPClose closes a connection to a fully encrypted channel.

## tSCPClose Standard properties

These properties are used to configure tSCPClose running in the Standard Job framework. 
The Standard tSCPClose component belongs to the Internet family. 
The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Component list</th>
<th>If there is more than one connection in the current Job, select tSCPConnection from the list.</th>
</tr>
</thead>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</th>
</tr>
</thead>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</th>
</tr>
</thead>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tSCPClose is generally used as a start component. It requires an output component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your SCP connection dynamically from multiple connections planned in your Job.</td>
</tr>
</tbody>
</table>
Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Related scenario

This component is closely related to **tSCPConnection** and **tSCPRollback**. It is generally used with **SCPConnection** as it allows you to close a connection for the transaction which is underway.

For a related scenario, see Scenario: handling a file using SCP on page 3492.
tSCPConnection

Opens an SCP connection to transfer files in one transaction.

**tSCPConnection Standard properties**

These properties are used to configure tSCPConnection running in the Standard Job framework.

The Standard tSCPConnection component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Host</th>
<th>IP address of the SCP server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of listening port of the SCP server.</td>
<td></td>
</tr>
<tr>
<td>User name for the SCP server.</td>
<td></td>
</tr>
<tr>
<td>SCP authentication method.</td>
<td></td>
</tr>
<tr>
<td>User password for the SCP server.</td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |
Usage

| Usage rule | This component is typically used as a single-component sub-job. It is used along with other SCP components. |

Related scenarios

For a related scenario, see Scenario: handling a file using SCP on page 3492.
tSCPDelete

Removes a file from the defined SCP server.
tSCPDelete deletes files from remote hosts over a fully encrypted channel.

**tSCPDelete Standard properties**

These properties are used to configure tSCPDelete running in the Standard Job framework.
The Standard tSCPDelete component belongs to the Internet family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>SCP IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of the SCP server.</td>
</tr>
<tr>
<td>Username</td>
<td>SCP user name.</td>
</tr>
<tr>
<td>Authentication method</td>
<td>SCP authentication method.</td>
</tr>
<tr>
<td>Password</td>
<td>SCP password.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Filelist</td>
<td>File name or path to the files to be deleted.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
<td></td>
</tr>
<tr>
<td>NB_FILE</td>
<td>the number of files processed. This is an After variable and it returns an integer.</td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>the execution result of the component. This is an After variable and it returns a string. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
<td></td>
</tr>
</tbody>
</table>
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

| Usage rule | This component is typically used as a single-component sub-job but can also be used with other components. |

### Related scenarios

No scenario is available for the Standard version of this component yet.
tSCPFileExists

Verifies the existence of a file on the defined SCP server.
tSCPFileExists checks, over a fully encrypted channel, if a file exists on a remote host.

**tSCPFileExists Standard properties**

These properties are used to configure tSCPFileExists running in the Standard Job framework.
The Standard tSCPFileExists component belongs to the Internet family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Host</th>
<th>SCP IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Listening port number of the SCP server.</td>
</tr>
<tr>
<td></td>
<td>SCP user name.</td>
</tr>
<tr>
<td></td>
<td>SCP authentication method.</td>
</tr>
<tr>
<td></td>
<td>SCP password.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td></td>
<td>File path on the remote directory.</td>
</tr>
<tr>
<td></td>
<td>Name of the file to check.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.
- **FILENAME**: the name of the file being processed. This is a Flow variable and it returns a string.
- **STATUS**: the execution result of the component. This is an After variable and it returns a string.
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide.*

### Usage

| Usage rule | This component is typically used as a single-component sub-job but can also be used with other components. |

### Scenario: handling a file using SCP

In this scenario, a six-component Job checks if a defined file exists in a specified directory on a remote SCP server, and returns a message to indicate the result of the verification. If the file doesn’t exist in that directory, the Job will put the local file to the SCP server, and then get the file from the SCP server to another defined local directory to verify that the file has been uploaded correctly.
Dropping and linking the components

Procedure

1. Add the following six components by typing their names on the design workspace or dropping them from the Palette to the design workspace: `tSCPConnection`, `tSCPFileExists`, `tJava`, `tSCPPut`, `tSCPGet`, and `tSCPClose`.
2. Connect the `tSCPConnection` component to the `tSCPFileExists` component using a Trigger > On Subjob Ok connection.
3. Connect the `tSCPFileExists` component to the `tJava` component using a Trigger > On Component Ok connection, and to the `tSCPPut` component using a Trigger > Run if connection.
4. Connect the `tSCPPut` component to the `tSCPGet` component using a Trigger > On Subjob Ok connection.
5. Connect the `tSCPGet` component to the `tSCPClose` component using a Trigger > On Subjob Ok connection.

Configuring the `tSCPConnection` component

Procedure

1. Double-click the `tSCPConnection` component to open its Basic settings view.

   ![tSCPConnection_1](image)

2. Enter the host name or IP address of the SCP server and the port number in the corresponding field.
3. Enter the user name used for connection in the Username field.
4. Select Password from the Authentication method drop-down list.
5. Click the [...] button next to the Password text box, then in the pop-up [Enter a new password] dialog box, enter the password between double quotes and click OK to save the change.
### Configuring file existence verification

**Procedure**

1. Double-click the **tSCPFileExists** component to open its Basic settings view.

![Basic settings view of tSCPFileExists component](image)

2. Select the *Use an existing connection* check box and then select the connection you have configured from the Component List.

3. In the *Filename* field, enter the name of the file to be checked between double quotes. In this example, it is "testscp.txt".

4. Double-click the **Java** component to open its Basic settings view.

![Basic settings view of tJava component](image)

5. In the Code field, enter the following code:

```java
System.out.println((String)globalMap.get("tSCPFileExists_1_FILENAME").
System.out.println((String)globalMap.get("tSCPFileExists_1_STATUS");
```

6. Select the Component tab and click the **If** connection to open its Basic settings view.

![Basic settings view of If component](image)

7. In the Condition field, enter the following code:

```java
!((String)globalMap.get("tSCPFileExists_1_STATUS").equals("File exists.")
```

This way, if the file doesn't exist in the defined directory, the rest of the Job will be executed.

### Configuring the tSCPPut component

**Procedure**

1. Double-click the **tSCPPut** component to open its Basic settings view.
Select the **Use an existing connection** check box and then select the connection you have configured from the **Component List**.

3. Click the [+ ] button to add one row in the **Filelist** table, and enter the full local path of the file to be uploaded in the corresponding **Source** column. In this example, it is "D:/SCPPut/testscp.txt".

### Configuring the tSCPGet component

**Procedure**

1. Double-click the **tSCPGet** component to open its **Basic settings** view.

2. Select the **Use an existing connection** check box and then select the connection you have configured from the **Component List**.

3. In the **Local directory** field, enter or click the [...] button to browse to the directory under which the file will be stored. In this example, it is "D:/SCPGet".

4. Select **append** from the **Overwrite or Append** drop-down list.

5. Click the [+ ] button to add one row in the **Filelist** table, and then enter the path of the file to be retrieved from the SCP server in the corresponding **Source** column. In this example, it is "/testscp.txt".

### Configuring the tSCPClose component

**Procedure**

1. Double-click the **tSCPClose** component to open its **Basic settings** view.
2. Select the connection you have configured from the **Component List**.

**Executing the Job**

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Make sure the file *testscp.txt* exists in the local directory *D:/SCPPut*, and does not exist in the local directory *D:/SCPGet* and the home directory of the SCP server.
3. Press **F6** to run the Job.

The file verification information is displayed on the **Run** console.

4. Go to the local directory *D:/SCPGet* and the home directory of the SCP server, you will see the file *testscp.txt* there.
tSCPFileList

Lists files from the defined SCP server.
tSCPFileList iterates, over a fully encrypted channel, on files in a given directory on a remote host.

**tSCPFileList Standard properties**

These properties are used to configure tSCPFileList running in the Standard Job framework. The Standard tSCPFileList component belongs to the Internet family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the host name or IP address of the SCP server.</td>
<td></td>
</tr>
<tr>
<td>Specify the listening port number of the SCP server.</td>
<td></td>
</tr>
<tr>
<td>Fill in the user name to access the SCP server.</td>
<td></td>
</tr>
<tr>
<td>Select one of the SCP authentication methods from the drop-down list, <strong>Public key</strong>, <strong>Password</strong>, or <strong>Keyboard Interactive</strong>.</td>
<td></td>
</tr>
<tr>
<td>Enter the password to access the SCP server.</td>
<td></td>
</tr>
<tr>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
<td></td>
</tr>
<tr>
<td>This field appears only when <strong>Password</strong> or <strong>Keyboard Interactive</strong> is selected from the <strong>Authentication method</strong> drop-down list.</td>
<td></td>
</tr>
<tr>
<td>Enter the passphrase for the public key.</td>
<td></td>
</tr>
<tr>
<td>To enter the passphrase, click the [...] button next to the passphrase field, and then in the pop-up dialog box enter the passphrase between double quotes and click <strong>OK</strong> to save the settings.</td>
<td></td>
</tr>
<tr>
<td>This field appears only when <strong>Public key</strong> is selected from the <strong>Authentication method</strong> drop-down list.</td>
<td></td>
</tr>
<tr>
<td>Browse to or type in the path to the private key.</td>
<td></td>
</tr>
<tr>
<td>This field appears only when <strong>Public key</strong> is selected from the <strong>Authentication method</strong> drop-down list.</td>
<td></td>
</tr>
<tr>
<td>Specify the character used to separate multiple commands.</td>
<td></td>
</tr>
</tbody>
</table>
Click the [+] button to add a command or a set of commands to list files in a given directory on the SCP server.

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>CURRENT_LINE: the file currently being iterated upon. This is a Flow variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>NB_LINE: the number of files iterated upon so far. This is a Flow variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>STATUS: the execution result of the component. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as a single-component sub-job or used with other components.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This component is particularly useful when used with tSCPGet to get multiple files in a directory given that tSCPGet does not support wildcarding.</td>
</tr>
</tbody>
</table>

**Related scenario**

For a related scenario using a different protocol, see *Listing and getting files/folders on an FTP directory* on page 1232.
tSCPGet

Copies files from the defined SCP server.
tSCPGet transfers defined files via an SCP connection over a fully encrypted channel.

**tSCPGet Standard properties**

These properties are used to configure tSCPGet running in the Standard Job framework.
The Standard tSCPGet component belongs to the Internet family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Host</th>
<th>SCP IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of the SCP server.</td>
</tr>
<tr>
<td>Username</td>
<td>SCP user name.</td>
</tr>
<tr>
<td>Authentication method</td>
<td>SCP authentication method.</td>
</tr>
</tbody>
</table>
| Password     | SCP password.                     
|              | To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Local directory | Path to the destination folder. |
| Overwrite or Append | List of available options for the transferred files. |
| Filelist     | File name or path to the file(s) to copy. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.
- **NB_FILE**: the number of files processed. This is an After variable and it returns an integer.
- **STATUS**: the execution result of the component. This is an After variable and it returns a string.
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

| Usage rule | This component is typically used as a single-component sub-job but can also be used with other components. |

### Related scenario

For a related scenario, see Scenario: handling a file using SCP on page 3492.
tSCPPut

Copies files to the defined SCP server.

tSCPPut copies defined files to a remote SCP server over a fully encrypted channel.

**tSCPPut Standard properties**

These properties are used to configure tSCPPut running in the Standard Job framework.
The Standard tSCPPut component belongs to the Internet family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>SCP IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of the SCP server.</td>
</tr>
<tr>
<td>Username</td>
<td>SCP user name.</td>
</tr>
<tr>
<td>Authentication method</td>
<td>SCP authentication method.</td>
</tr>
</tbody>
</table>
| Password            | SCP password.
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Remote directory    | Path. to the destination folder.                                           |
| Filelist            | File name or path to the file(s) to copy.                                  |

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE:</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td>NB_FILE:</td>
<td>the number of files processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>STATUS:</td>
<td>the execution result of the component. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press \texttt{Ctrl + Space} to access the variable list and choose the variable to use from it.

For further information about variables, see \textit{Talend Studio User Guide}.

### Usage

| Usage rule                                                                 | This component is typically used as a single-component sub-job but can also be used with other components. |

### Related scenario

For a related scenario, see Scenario: handling a file using SCP on page 3492.
tSCPRename

 Renames file(s) on the defined SCP server.

 **tSCPRename Standard properties**

 These properties are used to configure tSCPRename running in the Standard Job framework.

 The Standard tSCPRename component belongs to the Internet family.

 The component in this framework is generally available.

 **Basic settings**

<table>
<thead>
<tr>
<th>Host</th>
<th>SCP IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Listening port number of the SCP server.</td>
<td></td>
</tr>
<tr>
<td>SCP user name.</td>
<td></td>
</tr>
<tr>
<td>SCP authentication method.</td>
<td></td>
</tr>
<tr>
<td>SCP password.</td>
<td></td>
</tr>
<tr>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
<td></td>
</tr>
<tr>
<td>Enter the name or path to the file you want to rename.</td>
<td></td>
</tr>
<tr>
<td>Enter the file new name.</td>
<td></td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>STATUS: the execution result of the component. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
</tbody>
</table>
Usage

Usage rule
This component is typically used as a single-component sub-job but can also be used with other components.

Related scenario

For a related scenario using a different protocol, see Renaming a file located on an FTP server on page 1255.
tSCPTruncate

Removes data from file(s) on the defined SCP server via an SCP connection.

### tSCPTruncate Standard properties

These properties are used to configure tSCPTruncate running in the Standard Job framework. The Standard tSCPTruncate component belongs to the Internet family. The component in this framework is generally available.

#### Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>SCP IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of the SCP server.</td>
</tr>
<tr>
<td>Username</td>
<td>SCP user name.</td>
</tr>
<tr>
<td>Authentication method</td>
<td>SCP authentication method.</td>
</tr>
<tr>
<td>Password</td>
<td>SCP password.</td>
</tr>
<tr>
<td>Remote directory</td>
<td>Path to the destination file.</td>
</tr>
<tr>
<td>Filelist</td>
<td>File name or path to the file(s) to truncate.</td>
</tr>
</tbody>
</table>

#### Advanced settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
<tr>
<td>Statistics</td>
<td></td>
</tr>
</tbody>
</table>

#### Global Variables

- **ERROR MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.
- **NB_FILE**: the number of files processed. This is an After variable and it returns an integer.
- **STATUS**: the execution result of the component. This is an After variable and it returns a string.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press `Ctrl + Space` to access the variable list and choose the variable to use from it.
tSCPTruncate

For further information about variables, see *Talend Studio User Guide*.

**Usage**

| Usage rule | This component is typically used as a single-component sub-job but can also be used with other components. |

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tSendMail

Notifies recipients about a particular state of a Job or possible errors.
tSendMail sends emails and attachments to defined recipients.

tSendMail Standard properties

These properties are used to configure tSendMail running in the Standard Job framework.
The Standard tSendMail component belongs to the Internet family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>To</td>
<td>Main recipient email address.</td>
</tr>
<tr>
<td>From</td>
<td>Sending server email address.</td>
</tr>
<tr>
<td>Show sender's name</td>
<td>Select this check box if you want the sender name to show in the messages.</td>
</tr>
<tr>
<td>Cc</td>
<td>Email addresses of secondary recipients of the email message directed to another.</td>
</tr>
<tr>
<td>Bcc</td>
<td>Email addresses of secondary recipients of the email message. Recipients listed in the Bcc field receive a copy of the message but are not shown on any other recipient's copy.</td>
</tr>
<tr>
<td>Subject</td>
<td>Heading of the mail.</td>
</tr>
<tr>
<td>Message</td>
<td>Body message of the email. Press Ctrl+Space to display the list of available variables.</td>
</tr>
<tr>
<td>Die if the attachment file doesn't exist</td>
<td>This check box is selected by default. Clear this check box if you want the message to be sent even if there are no attachments.</td>
</tr>
<tr>
<td>Attachments</td>
<td>Specify the file(s) attached and sent along with the email message. Click the [+ button to add as many rows as needed and set the value of the following columns for each row:</td>
</tr>
<tr>
<td></td>
<td>• File: Enter the path to the attached file or the path to the folder if all files directly under it will be attached.</td>
</tr>
<tr>
<td></td>
<td>• Content Transfer Encoding: Click the cell and select a content transfer encoding value from the list.</td>
</tr>
<tr>
<td>Other Headers</td>
<td>Click the plus button to add as many lines as needed where you can type the Key and the corresponding Value of any header information that does not belong to the standard header.</td>
</tr>
<tr>
<td><strong>SMTP Host and Port</strong></td>
<td>IP address of SMTP server used to send emails.</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td><strong>SSL Support</strong></td>
<td>Select this check box to authenticate the server at the client side via an SSL protocol.</td>
</tr>
<tr>
<td><strong>STARTTLS Support</strong></td>
<td>Select this check box to authenticate the server at the client side via a STARTTLS protocol.</td>
</tr>
<tr>
<td><strong>Importance</strong></td>
<td>Select in the list the priority level of your messages.</td>
</tr>
</tbody>
</table>
| **Need authentication / Username and Password** | Select this check box and enter a username and a password in the corresponding fields if this is necessary to access the service.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| **Die on error**       | This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. |

### Advanced settings

| **MIME subtype from the ‘text’ MIME type** | Select in the list the structural form for the text of the message. |
| **Encoding type** | Select the encoding from the list or select **Custom** and define it manually. |
| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

### Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).
Usage

Usage rule

This component is typically used as one sub-job but can also be used as output or end object. It can be connected to other components with either Row or Iterate links.

Scenario: Email on error

This scenario creates a three-component Job which sends an email to defined recipients when an error occurs.

- Drop the following components from your Palette to the design workspace: tFileInputDelimited, tFileOutputXML, tSendMail.
- Define tFileInputDelimited properties. Related topic: tFileInputDelimited on page 1010.
- Right-click on the tFileInputDelimited component and select Row > Main. Then drag it onto the tFileOutputXML component and release when the plug symbol shows up.
- Define tFileOutputXML properties.
- Drag a Run on Error link from tFileDelimited to tSendMail component.
- Define the tSendMail component properties:
  - Enter the recipient and sender email addresses, as well as the email subject.
  - Enter a message containing the error code produced using the corresponding global variable. Access the list of variables by pressing Ctrl+Space.
  - Add attachments and extra header information if any. Type in the SMTP information.
In this scenario, the file containing data to be transferred to XML output cannot be found. *tSendmail* runs on this error and sends a notification email to the defined recipient.
tServerAlive

Validates the status of the connection to a specified host.

tServerAlive checkes whether the connection to a specified host is up or down, which is indicated by the boolean value returned by the global variable SERVER_ALIVE_RESULT.

**tServerAlive Standard properties**

These properties are used to configure tServerAlive running in the Standard Job framework.

The Standard tServerAlive component belongs to the Misc family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Select either Ping or Telnet from the drop-down list of protocols.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>IP address or host name of the remote server to be checked.</td>
</tr>
<tr>
<td>Timeout Interval (in seconds)</td>
<td>Enter the timeout interval in seconds. If the target host is reached within this time, the component’s global variable SERVER_ALIVE_RESULT will be set to true, otherwise false.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of the Telnet server. Available when Telnet is selected from the Protocol drop-down list.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

| Global Variables | SERVER_ALIVE_RESULT: the status of the connection to a remote host. This is an After variable and it returns a boolean.  
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is generally used as a one-component subjob.</th>
</tr>
</thead>
</table>

**Scenario: Validating the status of the connection to a remote host**

In this scenario, **tServerAlive** is used to ping a remote host and the value of its global variable *SERVER_ALIVE_RESULT* is extracted to show the connection status.

**Dropping and linking components**

**Procedure**

1. Drop the following components from the **Palette** onto the design workspace: **tServerAlive**, **tFixedFlowInput**, and **tLogRow**.
2. Connect **tServerAlive** to **tFixedFlowInput** using a **Trigger > On Subjob Ok** connection.
3. Connect **tFixedFlowInput** to **tLogRow** using a **Row > Main** connection.

**Configuring the components**

**Procedure**

1. Double-click **tServerAlive** to open its **Basic settings** view.
2. In the Protocol drop-down list, select **Ping**.
3. In the Host field, enter the IP address of the remote server to ping.
4. In the Timeout Interval field, enter the timeout interval in seconds.
5. Double-click **tFixedFlowInput** to open its Basic settings view.

6. Click the [...] button next to **Edit schema** to open the schema editor.

7. Click the [+•] button to add one column, namely **status**, of the boolean type. Click **OK** to validate the schema and close the editor.
   In the dialog box that appears, click **Yes** to propagate the changes to the subsequent component.
8. In the Values table, press the Ctrl + Space keys in the Value field to select **tServerAlive_1_SERVER_ALIVE_RESULT** from the variable list.
9. Double-click **tLogRow** to open its **Basic settings** view.

10. Select **Table (print values in cells of a table)** for a better display of the results.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Press **F6** to execute the Job.

```
[statistics] connecting to socket on port 3516
[statistics] connected

|tLogRow_1|
|status:
|true|
```

**Results**

As shown above, the remote server is pingable.
**tServiceNowConnection**

Opens a connection to a ServiceNow instance that can then be reused by other ServiceNow components.

**tServiceNowConnection Standard properties**

These properties are used to configure tServiceNowConnection running in the Standard Job framework.

The Standard tServiceNowConnection component belongs to the Business and the Cloud families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>URL</th>
<th>Enter the URL of the ServiceNow instance to be connected.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Username and Password</strong></td>
<td>Enter the user authentication data to connect to the ServiceNow instance.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>
Usage

| Usage rule | This component is more commonly used with other ServiceNow components. |

Related scenario

No scenario is available for the Standard version of this component yet.
**tServiceNowInput**

Accesses ServiceNow and retrieves data from it.

*tServiceNowInput* retrieves data from ServiceNow according to the specified conditions.

**tServiceNowInput Standard properties**

These properties are used to configure *tServiceNowInput* running in the Standard Job framework.

The Standard *tServiceNowInput* component belongs to the Business and the Cloud families.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use Existing Connection</th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td></td>
<td>For an example about how to share a database connection across Job levels, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>URL</strong></th>
<th>Enter the URL of the ServiceNow instance to be connected.</th>
</tr>
</thead>
</table>

| **Username and Password** | Enter the user authentication data to connect to the ServiceNow instance. |
|                          | To enter the password, click the […] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |

| **Table**                | Select the name of the table from which the data will be retrieved or select **Use Custom Table** and in the **Custom Table Name** field displayed enter the table name. |

| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the |
next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.

- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

### Conditions

Specify search conditions to retrieve only the data that matches the criteria by clicking the **[+]** button to add as many conditions as needed and then setting the following parameters for each condition:

- **Input column**: click in the cell and select the column of the input schema on which the search condition is to be set.

- **Operator**: click in the cell and select an operator for the search condition.

- **Value**: enter the value for the corresponding column for the search condition.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Trim all columns</strong></th>
<th>Select this check box to remove the leading and trailing whitespaces from all columns.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Limit</strong></td>
<td>Enter the maximum number of records to be retrieved.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
</table>
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

---

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is usually used as a start component in a <strong>Talend</strong> flow and it requires an output component.</th>
</tr>
</thead>
</table>

#### Dynamic settings

Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

---

### Related scenario

No scenario is available for the Standard version of this component yet.
tServiceNowOutput

Performs the defined action on the data on ServiceNow.
tServiceNowOutput inserts or updates data on ServiceNow.

**tServiceNowOutput Standard properties**

These properties are used to configure tServiceNowOutput running in the Standard Job framework. The Standard tServiceNowOutput component belongs to the Business and the Cloud families. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Use Existing Connection</strong></th>
<th>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td><strong>1.</strong> In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td><strong>2.</strong> In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td></td>
<td>For an example about how to share a database connection across Job levels, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>URL</strong></th>
<th>Enter the URL of the ServiceNow instance to be connected.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Username and Password</strong></th>
<th>Enter the user authentication data to connect to the ServiceNow instance.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Action</strong></th>
<th>Select an action to be performed on the data from the drop-down list:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Insert</strong>: insert data into ServiceNow.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update</strong>: update existing data in ServiceNow.</td>
</tr>
</tbody>
</table>
### Table

Select the name of the table to be written or select **Use Custom Table** and in the **Custom Table Name** field displayed enter the table name.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](#).
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](#).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content](#)** window.

### Advanced settings

<table>
<thead>
<tr>
<th>Enable payload debug mode</th>
<th>Select this check box to enable the payload debug mode and logging for the payload of the request message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable response debug mode</td>
<td>Select this check box to enable the response debug mode and logging for the response message from the log server.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong></th>
<th>the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED</strong></td>
<td>the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>NB_LINE_UPDATED</strong></td>
<td>the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>
**NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer.

**PAYLOAD**: the payload of the JSON request message. This is a Flow variable and it returns an object.

**RESPONSE**: the response message from the log server. This is a Flow variable and it returns an object.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

---

### Usage

<table>
<thead>
<tr>
<th><strong>Usage rule</strong></th>
<th>This component is usually used as an end component in a Talend flow and it requires an input component.</th>
</tr>
</thead>
</table>

#### Dynamic settings

Click the [+ ] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

---

### Related scenario

No scenario is available for the Standard version of this component yet.
tSetEnv

Adds variables temporarily to system environment during the execution of a Job.

tSetEnv creates variables and execute a Job script through communicating the information about the newly created variables between subjobs. After job execution, the newly created variables are deleted.

### tSetEnv Standard properties

These properties are used to configure tSetEnv running in the Standard Job framework.

The Standard tSetEnv component belongs to the System family.

The component in this framework is generally available.

#### Basic settings

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Click the plus button to add the variables needed for the job. name: Enter the syntax for the new variable. value: Enter a parameter value according to the context. append: Select this check box to add the new variable at the end.</th>
</tr>
</thead>
</table>

#### Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

#### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</th>
</tr>
</thead>
</table>

#### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tSetEnv can be used as a start or an intermediate component.</th>
</tr>
</thead>
</table>
Scenario: Modifying a variable during a Job execution

The following scenario is made of two Jobs parent and child. With the tSetEnv component, you can transfer and modify in a child Job a value created in a parent Job. As part of this Job, the tMsgBox components allow you to display, for information purposes only, that a variable is properly set, via an info-box.

To modify the value of the parent Job by using a variable set in the tSetEnv component, do as described in the following sections:

Drop and link components

Procedure

1. Create a first Job named parentJob: right-click on the Job Design node of the Repository, and choose Create a Job.
2. From the Palette, drop a tSetEnv component, two tMsgBox components, and one tRunJob component onto the design workspace.
3. Connect the tSetEnv component to a first tMsgBox component with a OnSubjobOk link : right-click on the start component, select Trigger, then OnSubjobOk. Then click on the end component you want to connect.
4. Connect the first tMsgBox component to the tRunJob with a OnSubjobOk link.
5. Then connect the tRunJob component to the second tMsgBox with a OnSubjobOk link.

![Diagram of Job design](image)

7. From the Palette, drop a tSetEnv component onto the design workspace.
8. Connect the tSetEnv component to the tMsgBox with a OnSubjobOk link: right-click on the start component, select Trigger, then OnSubjobOk. Then click on the end component you want to connect.
Set the components

About this task

In this example, the value set in the parent Job is transferred to the child Job. There, it is modified and adopts the value of the child Job, and then transferred to the parent Job again.

Procedure

1. In parentJob, select the tSetEnv component and click the Component tab. Add a variable row by clicking the [+ ] button to set the initial value of the variable. Type Variable_1 in the Name field, and Parent job value in the Value field.

2. Select the first tMsgBox component, and click the Component tab. In the Message field, type the message displayed in the info-box which confirms that your variable has properly been taken into account. For example: "Parent:"+System.getProperty("Variable_1") displays the variable set in the tSetEnv component (here Parent job value).

3. Select the second tMsgBox component, and click the Component tab. In the Message field, type the "Parent:"+System.getProperty("Variable_1") line again. It makes the variable set in the child Job appear.

4. Select the tRunJob component and click the Component tab. In the Job field, type the name of your child Job, here ChildJob. This will run the child Job when you run the parent Job.

5. Now double-click the tRunJob component to open the child Job ChildJob.
6. Select the **tSetEnv** component, and click the **Component** tab. Add a variable row by clicking the [+ button to set the initial value of the variable. Type **Variable_1** in the **Name** field, and **Child job value** in the **Value** field.

7. Select the **tMsgBox** component and click the **Component** tab. In the **Message** field, type the message displayed in the info-box which confirms that your variable has properly been taken into account. For example: "Son:"+System.getProperty("Variable_1") displays the variable set in the **tSetEnv** component (here **Child job value**).

<table>
<thead>
<tr>
<th>Title</th>
<th>&quot;Talend Open Studio&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Buttons</strong></td>
<td>OK</td>
</tr>
<tr>
<td><strong>Icon</strong></td>
<td>Icon Information</td>
</tr>
<tr>
<td><strong>Message</strong></td>
<td>&quot;Son:&quot;+System.getProperty(&quot;Variable_1&quot;)</td>
</tr>
</tbody>
</table>

8. Save your Job, go back to *parentJob*, then run the Job by pressing **F6**.

**Run the Job**

Three info-boxes are displayed one after the other:

- **Parent: Parent job value**: parent Job’s value is **Parent job value**.
- **Child: Child job value**: Child Job’s value is **Child job value**.
- **Parent: Child job value**: parent Job’s value was modified by the variable set in the **tSetEnv** of the child Job, then transferred again to the parent Job. parent Job’s value is now the one set in the child Job.
tSetGlobalVar

Facilitates the process of defining global variables.
tSetGlobalVar allows you to define and set global variables in GUI.

**tSetGlobalVar Standard properties**

These properties are used to configure tSetGlobalVar running in the Standard Job framework.
The Standard tSetGlobalVar component belongs to the Custom Code family.
The component in this framework is generally available.

**Basic settings**

| Variables | This table contains two columns.  
| Key: Name of the variable to be called in the code.  
| Value: Value assigned to this variable. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
| A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
| To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
| For further information about variables, see **Talend Studio User Guide**. |

**Usage**

| Usage rule | This component is generally used as a one-component subjob. |
| Limitation | Knowledge of Java language is required. |
**Scenario: Printing out the content of a global variable**

This scenario is a simple Job that prints out the value of a global variable defined in the `tSetGlobalVar` component.

**Setting up the Job**

**Procedure**

1. Drop the following components from the **Palette** onto the design workspace: `tSetGlobalVar` and `tJava`.
2. Connect the `tSetGlobalVar` component to the `tJava` component using a **Trigger > OnSubjobOk** connection.

**Configuring the tSetGlobalVar component**

**Procedure**

1. Double-click the `tSetGlobalVar` component to display its **Basic settings** view.

2. Click the plus button to add a line in the **Variables** table, and fill the **Key** and **Value** fields with `K1` and `20` respectively.

3. Then double-click the `tJava` component to display its **Basic settings** view.

Code:

```java
String foo = "bar";
String K1;
String Result = "The value is:";
Result = Result +
globalMap.get("K1");
System.out.println(Result);
```
4. In the **Code** area, type in the following lines:

```java
String foo = "bar";
String K1;
String Result = "The value is:";

Result = Result + globalMap.get("K1");

System.out.println(Result);
```

In this use case, we use the *Result* variable. To access the global variable list, press **Ctrl + Space** bar on your keyboard and select the relevant global parameter.

**Job execution**

Save your Job and press **F6** to execute it.

The content of global variable *K1* is displayed on the console.

```
Starting job SetGlobalVar at 16:22 25/02/2011
[statistics] connecting to socket on port 3489
[statistics] connected
The value is:20
[statistics] disconnected
Job SetGlobalVar ended at 16:22 25/02/2011.
[exit code=0]
```
tSetKerberosConfiguration

Sets the relevant information for Kerberos authentication.

tSetKerberosConfiguration is designed to configure Kerberos authentication for enhanced security of network communications.

For more information on the Kerberos protocol, go to [http://www.kerberos.org](http://www.kerberos.org).

### tSetKerberosConfiguration Standard properties

These properties are used to configure tSetKerberosConfiguration running in the Standard Job framework.

The Standard tSetKerberosConfiguration component belongs to the Internet family.

The component in this framework is generally available.

#### Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KDC Server</strong></td>
<td>Address of the Key Distribution Center (KDC) server.</td>
</tr>
<tr>
<td><strong>Realm</strong></td>
<td>Name of the Kerberos realm.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Kerberos authentication credentials. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>

#### Advanced settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

#### Global Variables

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>
**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is typically used as a sub-job by itself and is used along with <strong>tSoap</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td><strong>tSetKerberosConfiguration</strong> cannot work with IBM JVM.</td>
</tr>
</tbody>
</table>

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tSetKeystore

Sets the authentication data type between **PKCS 12** and **JKS**.

$tSetKeystore$ submits authentication data of a truststore with or without keystore to validation for the SSL connection.

### tSetKeystore Standard properties

These properties are used to configure $tSetKeystore$ running in the Standard Job framework.

The Standard $tSetKeystore$ component belongs to the Internet family.

The component in this framework is generally available.

#### Basic settings

<table>
<thead>
<tr>
<th><strong>TrustStore type</strong></th>
<th>Select the type of the TrustStore to be used. It may be <strong>PKCS 12</strong> or <strong>JKS</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TrustStore file</strong></td>
<td>Type in the path, or browse to the certificate TrustStore file (including filename) that contains the list of certificates that the client trusts.</td>
</tr>
<tr>
<td><strong>TrustStore password</strong></td>
<td>Type in the password used to check the integrity of the TrustStore data.</td>
</tr>
<tr>
<td><strong>Need Client authentication</strong></td>
<td>Select this check box to validate the keystore data. Once doing so, you need complete three fields:</td>
</tr>
<tr>
<td></td>
<td>- <strong>KeyStore type</strong>: select the type of the keystore to be used. It may be <strong>PKCS 12</strong> or <strong>JKS</strong>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>KeyStore file</strong>: type in the path, or browse to the file (including filename) containing the keystore data.</td>
</tr>
<tr>
<td></td>
<td>- <strong>KeyStore password</strong>: type in the password for this keystore.</td>
</tr>
<tr>
<td><strong>Check server identity</strong></td>
<td>Select this check box to make the Job verify the match between the hostname of the URL and the hostname of the server. If they mismatch, the verification mechanism asks whether this connection should be allowed.</td>
</tr>
</tbody>
</table>

#### Advanced settings

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

#### Global Variables

| **Global Variables** | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions |
only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used standalone.</th>
</tr>
</thead>
</table>

### Connections

**Outgoing links (from this component to another):**

- **Trigger:** Run if; On Subjob Ok, On Subjob Error, On Component Ok; On Component Error.

**Incoming links (from one component to this one):**

- **Trigger:** Run if, On Subjob Ok, On Component Ok, On Component Error.

For further information regarding connections, see *Talend Studio User Guide*.

### Limitation

n/a.

### Scenario: Extracting customer information from a private WSDL file

This scenario describes a three-component Job that connects to a private WSDL file in order to extract customer information.

The WSDL file used in this Job accesses the corresponding web service under the SSL protocol. For this purpose, the most relative code in this file reads as follows:

```xml
<wsdl:port name="CustomerServiceHttpSoap11Endpoint" binding="ns:CustomerServiceSoap11Binding">
</wsdl:port>
```

Accordingly, we enter the following code in the *server.xml* file of Tomcat:

```xml
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
  maxThreads="150" scheme="https" secure="true"
  clientAuth="true" sslProtocol="TLS"
  keystoreFile="D:/server.keystore" keystorePass="password"
  keystoreType="JKS"
  truststoreFile="D:/server.p12" truststorePass="password"
  truststoreType="PKCS12" />
```
So we need keystore files to connect to this WSDL file. To replicate this Job, proceed as follows:

- Drop the following components from the Palette onto the design workspace: tSetKeystore, tWebService, and tLogRow.

- Right-click tSetKeystore to open its contextual menu.
- In this menu, select Trigger > On Subjob Ok to connect this component to tWebService.
- Right-click tWebService to open its contextual menu.
- In this menu, select Row > Main to connect this component to tLogRow.
- Double-click tSetKeystore to open its Basic settings view and define the component properties.

- In the TrustStore type field, select PKCS12 from the drop-down list.
- In the TrustStore file field, browse to the corresponding truststore file. Here, it is server.p12.
- In the TrustStore password field, type in the password for this truststore file. In this example, it is password.
- Select the Need Client authentication check box to activate the keystore configuration fields.
- In the KeyStore type field, select JKS from the drop-down list.
- In the KeyStore file field, browse to the corresponding keystore file. Here, it is server.keystore.
- Double-click tWebService to open the component editor, or select the component in the design workspace and in the Basic settings view, click the three-dot button next to Service configuration.
In the **WSDL** field, browse to the private WSDL file to be used. In this example, it is `CustomerService.wsdl`.

Click the refresh button next to the **WSDL** field to retrieve the WSDL description and display it in the fields that follow.

In the **Port Name** list, select the port you want to use, `CustomerServiceHttpSoap11Endpoint` in this example.

In the **Operation** list, select the service you want to use. In this example the selected service is `getCustomer(parameters):Customer`.

Click **Next** to open a new view in the editor.

In the panel to the right of the **Input mapping** view, the input parameter of the service displays automatically. However, you can add other parameters if you select `[+ parameters]` and then click the
The Web service in this example has only one input parameter, ID.

- In the Expression column of the parameters.ID row, type in the customer ID of your interest between quotation marks. In this example, it is A00001.
- Click Next to open a new view in the editor.

In the Element list to the left of the view, the output parameter of the web service displays automatically. However, you can add other parameters if you select [+ parameters] and then click the plus button on top to display the [Parameter Tree] dialog box where you can select any of the parameters listed.

The Web service in this example has four output parameter: return.address, return.email, return.name and return.phone.

You now need to create a connection between the output parameter of the defined Web service and the schema of the output component. To do so:

- In the panel to the right of the view, click the three-dot button next to Edit Schema to open a dialog box in which you can define the output schema.
In the schema editing dialog box, click the plus button to add four columns to the output schema.

Click in each column and type in the new names, Name, Phone, Email and Address in this example. This will retrieve the customer information of your interest.

Click OK to validate your changes and to close the schema editing dialog box.

In the Element list to the right of the editor, drag each parameter to the field that corresponds to the column you have defined in the schema editing dialog box.

**Note:**

If available, use the Auto map! button, located at the bottom left of the interface, to carry out the mapping operation automatically.

Click OK to validate your changes and to close the editor.

In the design workspace, double-click tLogRow to open its Basic settings view and define its properties.

Click Sync columns to retrieve the schema from the preceding component.

Save your Job and press F6 to execute it.

The information of the customer with ID A00001 is returned and displayed in the console of Talend Studio.

```
<table>
<thead>
<tr>
<th>name</th>
<th>phone</th>
<th>email</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rose Gonzalez</td>
<td>(512) 757-9000</td>
<td><a href="mailto:rose@edge.com">rose@edge.com</a></td>
<td>313 Constitution Place Austin, TX 78767 USA</td>
</tr>
</tbody>
</table>
```

Job tSetKeystore ended at 19:01 24/11/2010. [exit code=0]
tSetProxy

Sets the relevant information for proxy setup.

**tSetProxy Standard properties**

These properties are used to configure `tSetProxy` running in the Standard Job framework.

The Standard `tSetProxy` component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxy type</td>
<td>List of proxy protocols.</td>
</tr>
<tr>
<td>Proxy host</td>
<td>Address of the proxy server.</td>
</tr>
<tr>
<td>Proxy port</td>
<td>Number of the proxy port.</td>
</tr>
<tr>
<td>Proxy user</td>
<td>Username for proxy authentication.</td>
</tr>
<tr>
<td>Proxy password</td>
<td>Password for proxy authentication.</td>
</tr>
<tr>
<td>None proxy hosts</td>
<td>The hosts which should be connected to directly and not through the proxy server. The value can be a list of hosts, each separated by a</td>
</tr>
</tbody>
</table>
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

| Usage rule          | Typically used as a sub-job by itself. **tSetProxy** is deployed along with other Internet components. |

### Related scenarios

No scenario is available for the Standard version of this component yet.
tSleep

Identifies possible bottlenecks using a time break in the Job for testing or tracking purpose.
tSleep implements a time off in a Job execution. In production, it can be used for any needed pause in the Job to feed input flow for example.

**tSleep Standard properties**

These properties are used to configure tSleep running in the Standard Job framework.
The Standard tSleep component belongs to the Orchestration family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Pause (in second)</th>
<th>Time in second the Job execution is stopped for.</th>
</tr>
</thead>
</table>

**Global Variables**

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |
|------------------|--------------------------------------------------------------------------------|

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>tSleep component is generally used as a middle component to make a break/pause in the Job, before resuming the Job.</th>
</tr>
</thead>
</table>
| Connections | Outgoing links (from this component to another):  
**Row:** Main; Iterate.  
**Trigger:** On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error  
Incoming links (from one component to this one):  
**Row:** Main; Reject; Iterate.  
**Trigger:** On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error; Synchronize; Parallelize. |
Related scenarios

For use cases in relation with tSleep, see Procedure on page 1992.
tSnowflakeClose

Closes an active Snowflake connection to release the occupied resources.

**tSnowflakeClose Standard properties**

These properties are used to configure tSnowflakeClose running in the Standard Job framework.

The Standard tSnowflakeClose component belongs to the Cloud family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Component</td>
<td>Select the component that opens the connection you need to close from the drop-down list.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

**Global variables**

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other Snowflake components, especially with the <strong>tSnowflakeConnection</strong> component.</th>
</tr>
</thead>
</table>

**Related scenario**

For a related scenario, see Writing data into and reading data from a Snowflake table on page 3548.
tSnowflakeConnection

Opens a connection to Snowflake that can then be reused by other Snowflake components.

**tSnowflakeConnection Standard properties**

These properties are used to configure tSnowflakeConnection running in the Standard Job framework. The Standard tSnowflakeConnection component belongs to the Cloud family. The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td>Account</td>
<td>In the <strong>Account</strong> field, enter, within double quotation marks, the name that has been assigned to you by Snowflake.</td>
</tr>
<tr>
<td>User Id and Password</td>
<td>Enter, within double quotation marks, your authentication information to log in Snowflake.</td>
</tr>
<tr>
<td></td>
<td>• In the <strong>User Id</strong> field, enter, within double quotation marks, your login name that has been defined in Snowflake using the LOGIN_NAME parameter of Snowflake. For details, ask the administrator of your Snowflake system.</td>
</tr>
<tr>
<td></td>
<td>• To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Warehouse</strong></td>
<td>Enter, within double quotation marks, the name of the Snowflake warehouse to be used. This name is case-sensitive and is normally upper case in Snowflake.</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Enter, within double quotation marks, the name of the database schema to be used. This name is case-sensitive and is normally upper case in Snowflake.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Enter, within double quotation marks, the name of the Snowflake database to be used. This name is case-sensitive and is normally upper case in Snowflake.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Login Timeout</strong></th>
<th>Specify how long to wait for a response when connecting to Snowflake before returning an error.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracing</strong></td>
<td>Select the log level for the Snowflake JDBC driver. If enabled, a standard Java log is generated.</td>
</tr>
<tr>
<td><strong>Role</strong></td>
<td>Enter, within double quotation marks, the default access control role to use to initiate the Snowflake session. This role must already exist and has been granted to the user ID you are using to connect to Snowflake. If this field is left empty, the PUBLIC role is automatically granted. For further information about the Snowflake access control model, see Snowflake documentation at <a href="#">Understanding the Access Control Model</a>.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global variables**

| **ERROR_MESSAGE** | The error message generated by the component when an error occurs. This is an After variable and it returns a string.                                                                                   |

**Usage**

| **Usage rule** | This component is more commonly used with other Snowflake components. It creates a Snowflake connection and works with other Subjobs that reuse this connection using a Trigger link. |

**Related scenario**

For a related scenario, see [Writing data into and reading data from a Snowflake table](#) on page 3548.
tSnowflakeInput

Reads data from a Snowflake table into the data flow of your Job based on an SQL query.

**tSnowflakeInput Standard properties**

These properties are used to configure tSnowflakeInput running in the Standard Job framework.

The Standard tSnowflakeInput component belongs to the Cloud family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in <strong>Repository &gt; Metadata</strong> will be reused by this component. You need to click the [...] button next to it and in the pop-up <strong>Repository Content</strong> dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the <strong>Connection Component</strong> drop-down list.</td>
</tr>
<tr>
<td><strong>Connection Component</strong></td>
<td>Select the component that opens the database connection to be reused by this component.</td>
</tr>
<tr>
<td><strong>Account</strong></td>
<td>In the <strong>Account</strong> field, enter, within double quotation marks, the name that has been assigned to you by Snowflake.</td>
</tr>
<tr>
<td><strong>User Id</strong> and <strong>Password</strong></td>
<td>Enter, within double quotation marks, your authentication information to log in Snowflake.</td>
</tr>
<tr>
<td></td>
<td>• In the <strong>User ID</strong> field, enter, within double quotation marks, your login name that has been defined in Snowflake using the <strong>LOGIN_NAME</strong> parameter of Snowflake. For details, ask the administrator of your Snowflake system.</td>
</tr>
</tbody>
</table>
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

**Warehouse**
Enter, within double quotation marks, the name of the Snowflake warehouse to be used. This name is case-sensitive and is normally upper case in Snowflake.

**Schema**
Enter, within double quotation marks, the name of the database schema to be used. This name is case-sensitive and is normally upper case in Snowflake.

**Database**
Enter, within double quotation marks, the name of the Snowflake database to be used. This name is case-sensitive and is normally upper case in Snowflake.

**Table**
Click the [...] button and in the displayed wizard, select the Snowflake table to be used.

**Schema and Edit Schema**
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.

- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.

If the Snowflake data type to be handled is VARIANT, OBJECT or ARRAY, while defining the schema in the component, select **String** for the corresponding data in the **Type** column of the schema editor wizard.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.

- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

Note that if the input value of any non-nullable primitive field is null, the row of data including that field will be rejected.

**Manual query**
Select this check box and in the **Full SQL query string** field, enter, within double quotation marks, the query...
statement to be executed. For more explanations about the Snowflake SQL commands, see SQL Command Reference.

| Condition | Enter, within double quotation marks, the boolean expression to be used to filter data from your Snowflake table. This expression is actually used as WHERE clause in the SELECT query run by tSnowflakeInput. |

**Advanced settings**

| Login Timeout | Specify how long to wait for a response when connecting to Snowflake before returning an error. |
| Tracing | Select the log level for the Snowflake JDBC driver. If enabled, a standard Java log is generated. |
| Role | Enter, within double quotation marks, the default access control role to use to initiate the Snowflake session. This role must already exist and has been granted to the user ID you are using to connect to Snowflake. If this field is left empty, the PUBLIC role is automatically granted. For further information about the Snowflake access control model, see Snowflake documentation at Understanding the Access Control Model. |
| Convert columns and table to uppercase | Select this check box to convert lowercase in the defined table name and schema column names to uppercase. This property is not available when the Manual Query check box is selected. |
| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| NB_LINE | The number of rows processed. This is an After variable and it returns an integer. |
| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

**Usage**

| Usage rule | This component is start component of a data flow in your Job. It sends data to other components via a Row link. |
Writing data into and reading data from a Snowflake table

Here is an example of using Talend Snowflake components to connect to Snowflake, create a Snowflake table, write data into the table, then read data from this table, finally close the connection to Snowflake.

Creating a Job for writing and reading Snowflake data

Create a Job to connect to Snowflake, create a Snowflake table, write data into the table and then read data from the table, finally close the connection to Snowflake.

Before you begin

- The Snowflake data warehouse to be used is started but not in the Suspended status.
- The Snowflake role assigned to you has read and write permissions to the database to be connected.

If you are not sure about these requirements, ask the administrator of your Snowflake system.

Procedure

1. In the Integration perspective of the Studio, create a new Job from the Job Designs node in the Repository tree view.
2. Add a `tSnowflakeConnection` component, a `tSnowflakeRow` component, a `tFixedFlowInput` component, a `tSnowflakeOutput` component, a `tSnowflakeInput` component, a `tLogRow` component, and a `tSnowflakeClose` component to the Job.

3. Connect the `tFixedFlowInput` component to the `tSnowflakeOutput` component using a Row > Main connection.

4. Do the same to connect the `tSnowflakeInput` component to the `tLogRow` component.

5. Connect the `tSnowflakeConnection` component to the `tSnowflakeRow` component using a Trigger > On Subjob Ok connection.

6. Do the same to connect the `tSnowflakeRow` component to the `tFixedFlowInput` component, the `tFixedFlowInput` component to the `tSnowflakeInput` component, and the `tSnowflakeInput` component to the `tSnowflakeClose` component.

## Connecting to Snowflake

Configure the `tSnowflakeConnection` component to connect to Snowflake.

**Procedure**

1. Double-click the `tSnowflakeConnection` component to open its Basic settings view.
2. In the Account field, enter the account name assigned by Snowflake.
3. In the User Id and the Password fields, enter the authentication information accordingly.
   
   Note that this user ID is your user login name. If you do not know your user login name yet, ask the administrator of your Snowflake system for details.
4. In the Warehouse field, enter the name of the data warehouse to be used in Snowflake.
5. In the Schema field, enter the name of the database schema to be used.
6. In the Database field, enter the name of the database to be used.

## Creating a Snowflake table

Configure the `tSnowflakeRow` component to create a new table.

**Procedure**

1. Double-click the `tSnowflakeRow` component to open its Basic settings view on the Component tab.
2. From the Connection Component drop-down list, select the `tSnowflakeConnection` component to reuse the connection created by it. In this example, it is `tSnowflakeConnection_1`.
3. In the Query field, enter the SQL statement used to create a new table into which the data will be written.
   
   In this example, the following SQL statement is used to create or replace a table `EMPLOYEE` with four columns, `ID` of `INT` type, `NAME` of `VARCHAR` type, `ONBOARD` of `DATE` type, and `SALARY` of `VARIANT` type.

   ```sql
   "CREATE OR REPLACE TABLE EMPLOYEE (" +
   "ID INT NOT NULL primary key," +
   "NAME VARCHAR (50), " +
   "ONBOARD DATE," +
   "SALARY VARIANT" +
   ") COMMENT = 'Created By Doc Team'"
   ```
Writing data into Snowflake

Configure the tFixedFlowInput component and the tSnowflakeOutput component to write data into Snowflake.

Procedure

1. Double-click the tFixedFlowInput component to open its Basic settings view.

2. Click the [...] button next to Edit schema and in the pop-up schema dialog box, define the schema by adding four columns, ID of int type, NAME and SALARY of String type, and ONBOARD of Date type.

3. Click OK to validate these changes and accept the propagation prompted by the pop-up dialog box.

4. In the Number of rows field, enter the number of records to be generated, 5 in this example.

5. In the Mode area, select Use Single Table and specify the value for each column.

   - ID: the automatically incremented number generated by the routine Numeric.sequence("id",1,1).
   - NAME: the random first name generated by the routine TalendDataGenerator.getFirstName().
   - ONBOARD: the random date generated by the routine TalendDate.getRandomDate("2007-09-01","2017-09-01").
• SALARY: the semi-structured JSON data with the format "{"Salary": value}, where the salary value is generated by the routine Numeric.random(100000,200000).

6. Double-click the tSnowflakeOutput component to open its Basic settings view.
7. From the Connection Component drop-down list, select the tSnowflakeConnection component to reuse the connection created by it.
8. Click the [...] button next to the Table field and in the pop-up dialog box, select the Use custom object check box and enter the name of the table into which the data will be written in the Object Name field. In this example, it is EMPLOYEE, the table created by the tSnowflakeRow component. When done, click OK to close the dialog box.
9. From the Output action drop-down list, select Insert to write data to the table EMPLOYEE.

Reading data from Snowflake

Configure the tSnowflakelnput component and the tLogRow component to retrieve data from Snowflake and output the data on the console.

Procedure

1. Double-click the tSnowflakelnput component to open its Basic settings view.
2. From the Connection Component drop-down list, select the tSalesforceConnection component to reuse the connection created by it.
3. Click the [...] button next to the Table field and in the pop-up dialog box, select the Use custom object check box and enter the name of the table from which the data will be retrieved in the Object Name field. In this example, it is EMPLOYEE, the table created by the tSnowflakeRow component. When done, click OK to close the dialog box.
4. Click the [...] button next to Edit schema and in the pop-up schema dialog box, define the schema by adding four columns, ID of int type, NAME and SALARY of String type, and ONBOARD of Date type. This schema is the same as the schema of the tSnowflakeOutput component.
5. Click OK to validate these changes and accept the propagation prompted by the pop-up dialog box.
6. Double-click the tLogRow component to open its Component view.
7. In the Mode area, select Vertical (each row is a key/value list) for better readability of the results.

Closing the Snowflake connection

Configure the tSnowflakeClose component to close the connection to Snowflake.

Procedure

1. Double-click the tSnowflakeClose component to open the Component tab.
2. From the Connection Component drop-down list, select the component that opens the connection you need to close, tSnowflakeConnection_1 in this example.

Executing the Job to write and read data in Snowflake

After setting up the Job and configuring the components used in the Job for writing and reading Snowflake data, you can then execute the Job and verify the Job execution result.
**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to run the Job.

As shown below, the data is written into the table `EMPLOYEE` in Snowflake, then read from this table and outputted on the console in the **Run** view of the Studio.

```
[statistics] connecting to socket on port 3880
[statistics] connected

#1. tLogRow_1
key | value
---+---
ID  | 1
NAME | John
ONBOARD | 2016-07-22
SALARY | {
  "Salary": 132039
}

#2. tLogRow_1
key | value
---+---
ID  | 2
NAME | Rutherford
ONBOARD | 2009-04-27
SALARY | {
  "Salary": 138064
}

#3. tLogRow_1
key | value
---+---
ID  | 3
NAME | Millard
ONBOARD | 2012-06-04
SALARY | {
  "Salary": 165287
}

#4. tLogRow_1
key | value
---+---
```
tSnowflakeOutput

Uses the data incoming from its preceding component to insert, update, upsert or delete data in a Snowflake table.

tSnowflakeOutput uses the bulk loader provided by Snowflake for high performance database operations.

**tSnowflakeOutput Standard properties**

These properties are used to configure tSnowflakeOutput running in the Standard Job framework.

The Standard tSnowflakeOutput component belongs to the Cloud family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
<tr>
<td>Connection Component</td>
<td>Select the component that opens the database connection to be reused by this component.</td>
</tr>
<tr>
<td>Account</td>
<td>In the Account field, enter, within double quotation marks, the name that has been assigned to you by Snowflake.</td>
</tr>
<tr>
<td>User Id and Password</td>
<td>Enter, within double quotation marks, your authentication information to log in Snowflake.</td>
</tr>
<tr>
<td></td>
<td>• In the User ID field, enter, within double quotation marks, your login name that has been defined in...</td>
</tr>
</tbody>
</table>
Snowflake using the LOGIN_NAME parameter of Snowflake. For details, ask the administrator of your Snowflake system.

- To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

<table>
<thead>
<tr>
<th>Warehouse</th>
<th>Enter, within double quotation marks, the name of the Snowflake warehouse to be used. This name is case-sensitive and is normally upper case in Snowflake.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>Enter, within double quotation marks, the name of the database schema to be used. This name is case-sensitive and is normally upper case in Snowflake.</td>
</tr>
<tr>
<td>Database</td>
<td>Enter, within double quotation marks, the name of the Snowflake database to be used. This name is case-sensitive and is normally upper case in Snowflake.</td>
</tr>
<tr>
<td>Table</td>
<td>Click the [...] button and in the displayed wizard, select the Snowflake table to be used.</td>
</tr>
</tbody>
</table>
| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
  **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.  
  **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.  
  If the Snowflake data type to be handled is VARIANT, OBJECT or ARRAY, while defining the schema in the component, select String for the corresponding data in the **Type** column of the schema editor wizard.  
  Click Edit schema to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
<table>
<thead>
<tr>
<th><strong>tSnowflakeOutput</strong></th>
<th>Note that if the input value of any non-nullable primitive field is null, the row of data including that field will be rejected.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output Action</strong></td>
<td>Select the operation to insert, delete, update or merge data in Snowflake tables. The <strong>Upsert</strong> operation allows you to merge data in a Snowflake table based on the data that is incoming to <strong>tSnowflakeOutput</strong>. After selecting <strong>Upsert</strong>, select the column to be used as the join key of this operation.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Login Timeout</strong></th>
<th>Specify how long to wait for a response when connecting to Snowflake before returning an error.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tracing</strong></td>
<td>Select the log level for the Snowflake JDBC driver. If enabled, a standard Java log is generated.</td>
</tr>
<tr>
<td><strong>Role</strong></td>
<td>Enter, within double quotation marks, the default access control role to use to initiate the Snowflake session. This role must already exist and has been granted to the user ID you are using to connect to Snowflake. If this field is left empty, the PUBLIC role is automatically granted. For further information about the Snowflake access control model, see Snowflake documentation at <a href="#">Understanding the Access Control Model</a>.</td>
</tr>
<tr>
<td><strong>Convert columns and table to uppercase</strong></td>
<td>Select this check box to convert lowercase in the defined table name and schema column names to uppercase. This property is not available when the <strong>Manual Query</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th><strong>NB_LINE</strong></th>
<th>The number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_SUCCESS</strong></td>
<td>The number of rows successfully processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_REJECT</strong></td>
<td>The number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>
### Usage

| Usage rule | This component is end component of a data flow in your Job. It receives data from other components via a **Row > Main** link. It can also send error messages to other components via a **Row > Rejects** link. The provided information about an error could be:  
- The name of the column in error.  
- The number of the row in error.  
- The category of the error, such as a parsing error or a conversion error.  
- The character offset in the line in error.  
- The related error message.  
- The offset of the byte in error.  
- The number of the line in error.  
- SQLSTATE for the error.  
- Snowflake error code. |

---

### Related scenario

For a related scenario, see [Writing data into and reading data from a Snowflake table](#) on page 3548.
tSnowflakeRow

Executes the SQL command stated onto a specified Snowflake database.

**tSnowflakeRow Standard properties**

These properties are used to configure tSnowflakeRow running in the Standard Job framework.

The Standard tSnowflakeRow component belongs to the Cloud family.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in. This property is not available when other connection component is selected from the Connection Component drop-down list.</td>
</tr>
<tr>
<td>Connection Component</td>
<td>Select the component that opens the database connection to be reused by this component.</td>
</tr>
<tr>
<td>Account</td>
<td>In the <strong>Account</strong> field, enter, within double quotation marks, the name that has been assigned to you by Snowflake.</td>
</tr>
<tr>
<td>User Id and Password</td>
<td>Enter, within double quotation marks, your authentication information to log in Snowflake.</td>
</tr>
<tr>
<td></td>
<td>• In the <strong>User ID</strong> field, enter, within double quotation marks, your login name that has been defined in Snowflake using the LOGIN_NAME parameter of Snowflake. For details, ask the administrator of your Snowflake system.</td>
</tr>
<tr>
<td></td>
<td>• To enter the password, click the [...] button next to the password field, and then in the pop-up dialog</td>
</tr>
</tbody>
</table>

3557
Enter the password between double quotes and click **OK** to save the settings.

**Warehouse**
Enter, within double quotation marks, the name of the Snowflake warehouse to be used. This name is case-sensitive and is normally upper case in Snowflake.

**Schema**
Enter, within double quotation marks, the name of the database schema to be used. This name is case-sensitive and is normally upper case in Snowflake.

**Database**
Enter, within double quotation marks, the name of the Snowflake database to be used. This name is case-sensitive and is normally upper case in Snowflake.

**Table**
Click the [...] button and in the displayed wizard, select the Snowflake table to be used.

**Schema and Edit Schema**
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields.

**Built-In:** You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository:** You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

If the Snowflake data type to be handled is VARIANT, OBJECT or ARRAY, while defining the schema in the component, select **String** for the corresponding data in the **Type** column of the schema editor wizard.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.

**Guess Query**
Click the button to generate the query which corresponds to the table and the schema in the **Query** field.

**Query**
Specify the SQL command to be executed.
**For more information about Snowflake SQL commands, see SQL Command Reference.**

| Die on error | Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row > Reject connection. |
| Advanced settings | Specify how long to wait for a response when connecting to Snowflake before returning an error. Select the log level for the Snowflake JDBC driver. If enabled, a standard Java log is generated. Enter, within double quotation marks, the default access control role to use to initiate the Snowflake session. This role must already exist and has been granted to the user ID you are using to connect to Snowflake. If this field is left empty, the PUBLIC role is automatically granted. For further information about the Snowflake access control model, see Snowflake documentation at Understanding the Access Control Model. Select this check box to propagate the result of the SELECT query to the output flow. Select this check box if you want to query the database using a prepared statement. In the Set PreparedStatement Parameters table displayed, specify the value for each parameter represented by a question mark ? in the SQL statement defined in the Query field. Specify the number of rows to be processed before committing batches of rows together into the database. Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

- **Login Timeout**
- **Tracing**
- **Role**
- **Propagate QUERYs recordset**
- **Use PreparedStatement**
- **Commit every**
- **tStatCatcher Statistics**
Dynamic settings

Dynamic settings
Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule | This component offers the flexibility of the database query and covers all possible SQL queries. |

Related scenario

For a related scenario, see Writing data into and reading data from a Snowflake table on page 3548.
tSOAP

Calls a method via a Web service in order to retrieve the values of the parameters defined in the component editor.

tSOAP sends the defined SOAP message with the given parameters to the invoked Web service and returns the value as defined, based on the given parameters.

**tSOAP Standard properties**

These properties are used to configure tSOAP running in the Standard Job framework.

The Standard tSOAP component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Schema and Edit schema</th>
<th>A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. This component always uses a built-in, read-only schema. By default, the schema contains three String type columns: - <strong>Header</strong>: stores the SOAP message header of the response from the server end. - <strong>Body</strong>: stores the SOAP message body of the response from the server end. - <strong>Fault</strong>: stores the error information when an error occurs during the SOAP message processing. If the <strong>Output in Document</strong> check box is selected, the schema then contains only one Document type column named <strong>Soap</strong>, which stores the whole response SOAP message in the XML format. Click <strong>Edit schema</strong> to view the schema structure.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use NTLM</td>
<td>Select this check box if you want to use the NTLM authentication protocol. <strong>Domain</strong>: Name of the client domain.</td>
</tr>
<tr>
<td>Need authentication</td>
<td>Select this check box and enter a user name and a password in the corresponding fields if this is necessary to access the service. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter...</td>
</tr>
<tr>
<td><strong>Use http proxy</strong></td>
<td>Select this check box if you are using a proxy server and fill in the necessary information.</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| **Trust server with SSL** | Select this check box to validate the server certificate to the client via an SSL protocol and fill in the corresponding fields:  
**TrustStore file:** enter the path (including filename) to the certificate TrustStore file that contains the list of certificates that the client trusts.  
**TrustStore password:** enter the password used to check the integrity of the TrustStore data. |
| **ENDPOINT** | Type in the URL address of the invoked Web server. |
| **SOAP Action** | Type in the URL address of the SOAPAction HTTP header field to be used to identify the intent of the SOAP HTTP request. |
| **SOAP version** | Select the version of the SOAP system you are using.  
**Warning:**  
The required SOAP Envelope varies among versions. |
| **Use a message from the input schema** | Select this check box to read a SOAP message from the preceding component to send to the invoked Web service.  
When this check box is selected, the SOAP message field becomes a drop-down list allowing you to select a Document type column to read an input XML file.  
**Warning:**  
This option makes sense only when the tSOAP component is connected with an input component the schema of which contains a Document type column to read a valid SOAP message. |
| **Output in Document** | Select this check box to output the response message in XML format. |
| **SOAP message** | Type in the SOAP message to be sent to the invoked Web service. The global and context variables can be used when you write a SOAP message.  
For further information about the context variables, see Talend Studio User Guide. |

**Advanced settings**

| **Use Kerberos** | Select this check box to choose a tSetKerberosConfiguration component from the Kerberos configuration list. |
Note:
The OnSubjobOk trigger of tSetKerberosConfiguration should be used for connection with tSoap.

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</th>
</tr>
</thead>
</table>

Global Variables

**Global Variables**

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**

This component can be used as an input or as an intermediate component.

**Connections**

Outgoing links (from this component to another):

- **Row**: Main; Iterate
- **Trigger**: Run if; On Component Ok; On Component Error.

Incoming links (from one component to this one):

- **Row**: Main; Iterate
- **Trigger**: Run if; On Component Ok; On Component Error.

For further information regarding connections, see *Talend Studio User Guide*.

**Scenario 1: Fetching the country name information using a Web service**

This scenario describes a two-component Job that uses a Web service to retrieve the country name information of a given country code.
Procedure

1. Drop the following components from the Palette onto the design workspace: tSOAP and tLogRow.

![Diagram of tSOAP and tLogRow components connected]

2. Connect tSOAP to tLogRow using a Row > Main link.

3. Double-click tSOAP to open its Basic settings view and define the component properties.

4. In **ENDPOINT** field, type in or copy-paste the URL address of the Web service to be used between the quotation marks: "http://www.webservicex.net/country.asmx".

5. In the **SOAP Action** field, type in or copy-paste the URL address of the SOAPAction HTTP header field that indicates that you want to retrieve the country name information: http://www.webserviceX.NET/GetCountryByCountryCode.

**Note:**

You can see this address by looking at the WSDL for the Web service you are calling. For the Web service of this example, in a web browser, append ?wsdl on the end of the URL of the Web service used in the **ENDPOINT** field, open the corresponding web page, and then see the SOAPAction defined under the operation node:

```xml
<wSDL:operation name="GetCountryByCountryCode">
```
6. From the **SOAP Version** list, select the version of the SOAP system being used. In this scenario, the version is **SOAP 1.1**.

7. In the **SOAP message** field, enter the XML-format message used to retrieve the country name information from the invoked Web service. In this example, `IS` is used as the country code, so the message is:

   ```xml
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:web="http://www.webserviceX.NET">
    <soapenv:Header/>
    <soapenv:Body>
    <web:GetCountryByCountryCode>
      <!--Optional:-->
    </web:GetCountryByCountryCode>
    </soapenv:Body>
   </soapenv:Envelope>
   ```

8. Save your Job and press **F6** to execute it.

   The country name information of the given country code `IS` is returned and displayed on the console of the **Run** view.

---

**Scenario 2: Using a SOAP message from an XML file to get country name information and saving the information to an XML file**

This scenario describes a three-component Job that uses a SOAP message from an input XML file to invoke a Web service for the country name information which corresponds to the country code `IR`, and writes the response to an XML file.

**Dropping and linking the components**

**Procedure**

1. Drop the following components from the **Palette** onto the design workspace: `tFileInputXML`, `tSOAP`, and `tFileOutputXML`.

2. Connect the components using **Row > Main** links.
Configuring the input component

Procedure

1. Double-click the tFileInputXML component to open its Basic settings view.

![tFileInputXML Component](image)

2. Click the [...] button next to Edit schema to open the [Schema] dialog box.

![Schema Dialog Box](image)

3. Click the [+] button to add a column, give it a name, getCountryName in this example, and select Document from the Type list. Then, click OK to close the dialog box.

4. In the File name/Stream field, enter the path to the input XML file that contains the SOAP message to be used, or browse to the path by clicking the [...] button.

The input file contains the following SOAP message and you can see that the given country code is IR:

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:web="http://www.webserviceX.NET">
  <soapenv:Header/>
  <soapenv:Body>
    <web:GetCountryByCountryCode>
      <web:CountryCode>IR</web:CountryCode>
    </web:GetCountryByCountryCode>
  </soapenv:Body>
</soapenv:Envelope>
```

5. In the Loop XPath query field, enter "/" to define the root as the loop node of the input file structure.
6. In the Mapping table, fill the XPath query column with "." to extract all data from context node of the source, and select the Get Nodes check box to build a Document type data flow.

Configuring the Web service via the tSOAP component

Procedure

1. Double-click the tSOAP component to open its Basic settings view.

2. In ENDPOINT field, enter or copy-paste the URL address of the Web service to be used between the quotation marks: “http://www.webservicex.net/country.asmx”.

3. In the SOAP Action field, enter or copy-paste the URL address of the SOAPAction HTTP header field that indicates that you want to retrieve the country name information: http://www.webserviceX.NET/GetCountryByCountryCode.

4. Select the Use a message from the schema check box, and select a Document type column from the SOAP Message list to read the SOAP message from the input file to send to the Web service. In this example, the input schema has only one column, getCountryName.

5. Select the Output in Document check box to output the response message in XML format.

Configuring the output component

Procedure

1. Double-click the tFileOutputXML component to open its Basic settings view.

2. In the File Name field, enter the path to the output XML file.

3. Select the Incoming record is a document check box to retrieve the incoming data flow as an XML document. Note that a Column list appears allowing you choose a column to retrieve data from. In this example, the schema contains only one column.
Executing the Job

Procedure

1. Press **Ctrl+S** to save your Job.
2. Press **F6**, or click **Run** on the **Run** tab to execute the Job.

The country name information that corresponds to the country code of *IR* is returned and the information is saved in the defined XML file.

```xml
<?xml version="1.0" encoding="ISO-8859-15"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    <soap:Body>
            <GetCountryByCountryCodeResult>&lt;NewDataSet&gt; &lt;Table&gt;
                &lt;countrycode&gt;IR&lt;/countrycode&gt; &lt;name&gt;Iran&lt;/name&gt;
                &lt;/Table&gt;
            &lt;/NewDataSet&gt;</GetCountryByCountryCodeResult>
        </GetCountryByCountryCodeResponse>
    </soap:Body>
</soap:Envelope>
```
tSocketInput

Opens the socket port and listens for the incoming data.

tSocketInput is a listening component, allowing to pass data via a defined port.

**tSocketInput Standard properties**

These properties are used to configure tSocketInput running in the Standard Job framework.
The Standard tSocketInput component belongs to the Internet family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host name</td>
<td>Name or IP address of the Host server</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port to open</td>
</tr>
<tr>
<td>Timeout</td>
<td>Number of seconds for the port to listen before closing.</td>
</tr>
<tr>
<td>Uncompress</td>
<td>Select this check box to unzip the data if relevant</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
<tr>
<td>Field separator</td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td>Row separator</td>
<td>String (ex: <code>\n</code> on Unix) to distinguish rows.</td>
</tr>
<tr>
<td>Escape Char</td>
<td>Character of the row to be escaped</td>
</tr>
<tr>
<td>Text enclosure</td>
<td>Character used to enclose text.</td>
</tr>
</tbody>
</table>

**Schema type** and **Edit Schema**

A schema is a row description, that is to say, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No**.
upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in:** The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide.

**Repository:** The schema already exists and is stored in the Repository, hence can be reused in various projects and job flowcharts. Related topic: see Talend Studio User Guide.

**Encoding type**
Select the encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling.

**Advanced settings**

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

**Global Variables**

**Global Variables**

**ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

**NB_LINE:** the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

**Usage**

**Usage rule**
This component opens a point of access to a workstation or server. This component starts a Job and only stops after the time goes out.

**Limitation**
Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find
Scenario: Passing on data to the listening port

The following scenario describes two Jobs aiming at passing data via a listening port. The first Job (SocketInput) opens the listening port and waits for the data to be sent over. The second Job (SocketOutput) passes delimited data from a file to a defined port number corresponding to the listening port.

Another application for the Socket components would be to allow controlled communication between servers which cannot communicate directly.

Dropping and linking components

Procedure
1. For the first Job, drop a tSocketInput component and a tLogRow component from the Palette to the design workspace, and link them using a Row > Main connection.
2. For the second Job, drop a tFileInputDelimited component and a tSocketOutput component from the Palette to the design workspace, and link them using a Row > Main connection.

Configuring the Jobs

Procedure
1. On the second Job, select the tFileInputDelimited and on the Basic Settings tab of the Component view, set the access parameters to the input file.
2. In File Name, browse to the file, and fill the Row, Field separators, and Header fields according to the input file used.

3. Describe the Schema of the data to be passed on to the tSocketOutput component.

The schema should be propagated automatically to the output component.

4. Select the tSocketOutput component and set the parameters on the Basic Settings tab of the Component view.

5. Define the Host IP address and the Port number where the data will be passed on to.

6. Set the number of retries in the Retry field and the amount of time (in seconds) after which the Job will time out.

7. Now on the other Job (SocketInput) design, define the parameters of the tSocketInput component.

8. Define the Host IP address and the listening Port number where the data are passed on to.

9. Set the amount of time (in seconds) after which the Job will time out.
10. Edit the schema and set it to reflect the whole or part of the other Job's schema.

**Executing the Jobs**

**Procedure**

1. Press **F6** to execute this Job (SocketInput) first, in order to open the listening port and prepare it to receive the passed data.

2. Before the time-out, launch the other Job (SocketOutput) to pass on the data.

   The result displays on the **Run** view, along with the opening socket information.

---

*Starting job SocketInput at 17:53 04/02/2008*

socket connected

AL|Alabama|Montgomery|Birmingham
AZ|Alaska|Juneau|Anchorage
AR|Arkansas|Little Rock|Little Rock
CA|California|Sacramento|Los Angeles
CO|Colorado|Denver|Denver
CT|Connecticut|Hartford|Bridgeport
DE|Delaware|Dover|Wilmington
FL|Florida|Tallahassee|Jacksonville
GA|Georgia|Atlanta|Atlanta
HI|Hawaii|Honolulu|Honolulu
ID|Idaho|Boise|Boise
IL|Illinois|Springfield|Chicago
IN|Indiana|Indianapolis|Indianapolis
tSocketOutput

Sends out the data from the incoming flow to a listening socket port.
The tSocketOutput component writes data to a listening port.

**tSocketOutput Standard properties**

These properties are used to configure tSocketOutput running in the Standard Job framework.
The Standard tSocketOutput component belongs to the Internet family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host name</td>
<td>Name or IP address of the Host server</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port to open</td>
</tr>
<tr>
<td>Compress</td>
<td>Select this check box to zip the data if relevant.</td>
</tr>
<tr>
<td>Retry times</td>
<td>Number of retries before the Job fails.</td>
</tr>
<tr>
<td>Timeout</td>
<td>Number of seconds for the port to listen before closing.</td>
</tr>
<tr>
<td>Die on error</td>
<td>Clear this check box to skip the row on error and complete the process for error-free rows.</td>
</tr>
<tr>
<td>Field separator</td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td>Row separator</td>
<td>String (ex: &quot;\n&quot; on Unix) to distinguish rows.</td>
</tr>
<tr>
<td>Escape Char</td>
<td>Character of the row to be escaped</td>
</tr>
<tr>
<td>Text enclosure</td>
<td>Character used to enclose text.</td>
</tr>
</tbody>
</table>

**Schema and Edit Schema**

A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No.
upon completion and choose this schema metadata again in the [Repository Content] window.

| **Built-in**: The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide. |
| **Repository**: The schema already exists and is stored in the Repository, hence can be reused in various projects and job flowcharts. Related topic: see Talend Studio User Guide. |

**Encoding**
- Select the encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling.

**Global Variables**

| **Global Variables** | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |
| **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer. |

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

**Usage**

| **Usage rule** | This component opens a point of access to a workstation or server. This component starts a Job and only stops after the time goes out. |

| **Limitation** | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |
Related Scenario

For use cases in relation with `tSocketOutput`, see `Scenario: Passing on data to the listening port` on page 3571
tSortRow

Helps creating metrics and classification table. tSortRow sorts input data based on one or several columns, by sort type and order.

**tSortRow Standard properties**

These properties are used to configure tSortRow running in the Standard Job framework.

The Standard tSortRow component belongs to the Processing family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
• View schema: choose this option to view the schema only.  
• Change to built-in property: choose this option to change the schema to Built-in for local changes.  
• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.  
Click Sync columns to retrieve the schema from the previous component connected in the Job. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Click + to add as many lines as required for the sort to be complete. By default the first column defined in your schema is selected.</td>
</tr>
<tr>
<td>Schema column: Select the column label from your schema, which the sort will be based on. Note that the order is essential as it determines the sorting priority.</td>
<td></td>
</tr>
</tbody>
</table>
**Sort type:** Numerical and Alphabetical order are proposed. More sorting types to come.

**Order:** Ascending or descending order.

**Advanced settings**

**Sort on disk**
- Customize the memory used to temporarily store output data.
- **Temp data directory path:** Set the location where the temporary files should be stored.
- **Create temp data directory if not exists:** Select this check box to create the directory if it does not exist.
- **Buffer size of external sort:** Type in the size of physical memory you want to allocate to sort processing.

**tStatCatcher Statistics**
- Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

**Global Variables**
- **ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.
- A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
- To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
- For further information about variables, see *Talend Studio User Guide*.

**Usage**

**Usage rule**
- This component handles flow of data therefore it requires input and output, hence is defined as an intermediary step.

**Sorting entries**

This scenario describes a three-component Job. A **tRowGenerator** is used to create random entries which are directly sent to a **tSortRow** to be ordered following a defined value entry. In this scenario, we suppose the input flow contains names of salespersons along with their respective sales and their years of presence in the company. The result of the sorting operation is displayed on the **Run** console.
• Drop the three components required for this use case: `tRowGenerator`, `tSortRow` and `tLogRow` from the Palette to the design workspace.

• Connect them together using Row main links.

• On the `tRowGenerator` editor, define the values to be randomly used in the Sort component. For more information regarding the use of this particular component, see `tRowGenerator` on page 3281.

• In this scenario, we want to rank each salesperson according to its Sales value and to its number of years in the company.

• Double-click `tSortRow` to display the Basic settings tab panel. Set the sort priority on the Sales value and as secondary criteria, set the number of years in the company.

• Use the plus button to add the number of rows required. Set the type of sorting, in this case, both criteria being integer, the sort is numerical. At last, given that the output wanted is a rank classification, set the order as descending.

• Display the Advanced Settings tab and select the Sort on disk check box to modify the temporary memory parameters. In the Temp data directory path field, type the path to the directory where you want to store the temporary data. In the Buffer size of external sort field, set the maximum buffer value you want to allocate to the processing.

⚠️ Warning:
The default buffer value is 1000000 but the more rows and/or columns you process, the higher the value needs to be to prevent the Job from automatically stopping. In that event, an "out of memory" error message displays.

- Make sure you connected this flow to the output component, `tLogRow`, to display the result in the Job console.

- Press `F6` to run the Job. The ranking is based first on the Sales value and then on the number of years of experience.
**tSplitRow**

Splits one input row into several output rows.

**tSplitRow Standard properties**

These properties are used to configure tSplitRow running in the Standard Job framework.

The Standard tSplitRow component belongs to the Processing family.

The component in this framework is generally available.

### Basic settings

| **Schema and Edit Schema** | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.  
Click **Sync columns** to retrieve the schema from the previous component connected in the Job. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>The schema will be created and stored locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>The schema already exists and is stored in the Repository, hence can be reused in various projects and Job flowcharts. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td><strong>Columns mapping</strong></td>
<td>Click the plus button to add as many lines as needed by mappings from input columns onto output columns.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th><strong>tStatCatcher Statistics</strong></th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-------------------------------</td>
<td>------------------------------------------------</td>
</tr>
</tbody>
</table>
Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule | This component splits one input row into multiple output rows by mapping input columns onto output columns. |

Scenario 1: Splitting one row into two rows

This scenario describes a three-component Job. A row of data containing information of two companies will be split up into two rows.

Procedure

Procedure

1. Drop the following components required for this use case: tFixedFlowInput, tSplitRow and tLogRow from the Palette to the design workspace.
2. Connect them together using Row Main connections.
3. Double-click tFixedFlowInput to open its Basic settings view.
4. Select **Use Inline Content(delimited file)** in the **Mode** area.

5. Fill the **Content** area with the following scripts:

   Talend;LA;California;537;5thAvenue;IT;Lionbridge;Memphis;Tennessee;537;Lincoln Road;IT Service;

6. Click **Edit schema** to open a dialog box to edit the schema for the input data.

7. Click the plus button to add twelve lines for the input columns: **Company, City, State, CountryCode, Street, Industry, Company2, City2, State2, CountryCode2, Street2** and **Industry2**.

8. Click **OK** to close the dialog box.

9. Double-click **tSplitRow** to open its **Basic settings** view.
10. Click **Edit schema** to set the schema for the output data.

11. Click the plus button beneath the **tSplitRow_1(Output)** table to add four lines for the output columns: **Company**, **CountryCode**, **Address** and **Industry**.

12. Click **OK** to close the dialog box. Then an empty table with column names defined in the preceding step will appear in the **Columns mapping** area:

<table>
<thead>
<tr>
<th>Column</th>
<th>Key</th>
<th>Type</th>
<th>✓</th>
<th>N..</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>City</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>State</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>CountryCode</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Street</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Industry</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Company2</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>City2</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>State2</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>CountryCode2</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Street2</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Industry2</td>
<td></td>
<td>String</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

13. Click the plus button beneath the empty table in the **Columns mapping** area to add two lines for the output rows.

14. Fill the table in the **Columns mapping** area by columns with the following values:
   - **Company**: row1.Company, row1.Company2;
Industry: row1.Industry, row1.Industry2;

Columns mapping

<table>
<thead>
<tr>
<th>Company</th>
<th>CountryCode</th>
<th>Address</th>
<th>Industry</th>
</tr>
</thead>
</table>

Note:
The value in Address column, for example, row1.Street+","+row1.City+","+row1.State, will display an absolute address by combining values in Street column, City column and State column together. The "row1" used in the values of each column refers to the input row from tFixedFlowInput.

15. Double-click tLogRow to open its Basic settings view.

16. Click Sync columns to retrieve the schema defined in the preceding component.

17. Select Table in the Mode area.

18. Save the Job and press F6 to run it.

Results

[statistics] connecting to socket on port 3706
[statistics] connected

<table>
<thead>
<tr>
<th>Company</th>
<th>CountryCode</th>
<th>Address</th>
<th>Industry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teldor</td>
<td>537</td>
<td>5thAvenue, LA, California</td>
<td>IT</td>
</tr>
<tr>
<td>Lionbridge</td>
<td>537</td>
<td>Lincoln Road, Memphis, Tennessee</td>
<td>IT Service</td>
</tr>
</tbody>
</table>

[statistics] disconnected
Job Split ended at 16:21 27/10/2011. [exit code=0]

The input data in one row is split into two rows of data containing the same company information.
**tSplunkEventCollector**

Sends the event data to Splunk through Splunk HTTP Event Collector.

`tSplunkEventCollector` collects and sends the event data to Splunk.

**tSplunkEventCollector Standard properties**

These properties are used to configure `tSplunkEventCollector` running in the Standard Job framework. The Standard `tSplunkEventCollector` component belongs to the Business Intelligence family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Schema and Edit schema</strong></th>
<th>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

Note that the schema of this component has been set by default with the following fields. You can click the [...] button next to **Edit schema** to view and change the predefined schema.

- **time**: the event time. Note that the input data is in Java Date format, and it will be transformed to the epoch time format required by Splunk before sending to Splunk HTTP Event Collector.
- **source**: the source value of the event data. It is usually the file or directory path, network port, or script from which the event originated.
- **sourcetype**: the source type of the event data. It tells what kind of data it is.
- **host**: the host of the event data. It is usually the host name, IP address, or fully qualified domain name of the network machine from which the event originated.
- **index**: the name of the index by which the event data is to be indexed. It must be within the list of allowed indexes if the token has the `indexes` parameter set.

For more information about the format of the event data sent to Splunk HTTP Event Collector, see *About the JSON event protocol in HTTP Event Collector*. 
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.

- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Splunk Server URL</th>
<th>Enter the URL used to access the Splunk Web Server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token</td>
<td>Specify the Event Collector token used to authenticate the event data. For more information, see <a href="#">HTTP Event Collector token management</a>.</td>
</tr>
</tbody>
</table>

### Advanced settings

**Extended output**

Select this check box to send the event data to Splunk in batch mode. In the field displayed, enter the number of events to be processed in each batch.

By default, this check box is selected and the number of events to be processed in each batch is 100.

| tStatCatcher Statistics      | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

### Global Variables

**Global Variables**

- **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

- **RESPONSE_CODE**: the response code from Splunk. This is an After variable and it returns an integer.

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).
## Usage

| Usage rule | This component is usually used as an end component of a Job or Subjob and it always needs an input link. |

### Related scenario

No scenario is available for this component yet.
tSPSSInput

Addresses SPSS .sav data to write it for example in another file.
tSPSSInput reads data from an SPSS .sav file.

**tSPSSInput Standard properties**

These properties are used to configure tSPSSInput running in the Standard Job framework.
The Standard tSPSSInput component belongs to the Business Intelligence family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sync schema</td>
<td>Click this button to synchronize with the columns of the input SPSS .sav file.</td>
</tr>
</tbody>
</table>
| **Schema and Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
The schema metadata in this component is retrieved directly from the input SPSS .sav file and thus is read-only.
You can click **Edit schema** to view the retrieved metadata. |
| Filename                 | Name or path of the SPSS .sav file to be read.                                                                                                                                                              |
| Translate labels         | Select this check box to translate the labels of the stored values.                                                                                                                                           |

**Advanced settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**

This component is used as a start component. It requires an output flow.

**Warning:**

Before being able to benefit from all functional objectives of the SPSS components, make sure to do the following:
- If you have already installed SPSS, add the path to the SPSS directory as the following: `SET PATH=%PATH%;<DR>:\program\SPSS`, or
- If you have not installed SPSS, you must copy the SPSS IO “spssio32.dll” lib from the SPSS installation CD and paste it in *Talend* root directory.

**Limitation**

Oracle provides two kinds of JVM platforms (32-bit and 64-bit). By default, the JVM used in a 64-bit operating system is the 64-bit version of the JVM. Since the JSPSS.dll file used by this component is compiled in the 32-bit JVM, it is needed to configure the 32-bit JVM for Job execution in *Talend Studio* installed on a 64-bit operating system.

To do so, perform the following:

1. Open the relevant Job in the workspace.
2. Click the **Run** tab and enter the **Advanced settings** view.
3. Select the **Use specific JVM arguments** check box and click the **New...** button.
4. In the **Set the VM argument** box, enter `-d32` and click **Ok** for validation.

Scenario: Displaying the content of an SPSS .sav file

The following scenario creates a two-component Job, which aims at reading each row of a .sav file and displaying the output on the log console.

Setting up the Job

**Procedure**

1. Drop a **tSPSSInput** component and a **tLogRow** component from the **Palette** onto the design workspace.
2. Right-click on tSPSSInput and connect it to tLogRow using a Main Row link.

**Configuring the input component**

**Procedure**

1. Click tSPSSInput to display its Basic settings view and define the component properties.

2. Click the three-dot button next to the Filename field and browse to the SPSS .sav file you want to read.

3. Click the three-dot button next to Sync schema. A message opens up prompting you to accept retrieving the schema from the defined SPSS file.

4. Click Yes to close the message and proceed to the next step.

5. If required, click the three-dot button next to Edit schema to view the pre-defined data structure of the source SPSS file.
6. Click **OK** to close the dialog box.

**Job execution**

Save the Job and press **F6** to execute it.

The SPSS file is read row by row and the extracted fields are displayed on the log console.

```
Starting job SPSS at 10:28 20/01/2010.
[statistics] connecting to socket on port 3802
[statistics] connected
1.0 1.0 57997.0 198522.0
2.0 1.0 57113.0 203200.0
3.0 1.0 54123.0 200744.0
1.0 0.0 63936.0 197791.0
2.0 0.0 64835.0 195714.0
3.0 0.0 66804.0 208239.0
[statistics] disconnected
Job SPSS ended at 10:28 20/01/2010. [exit code=0]
```

**Translating the stored values**

**About this task**

To translate the stored values, complete the following:

**Procedure**

1. In the **Basic settings** view, select the **Translate label** check box.
2. Click **Sync Schema** a second time to retrieve the schema after translation.
   - A message opens up prompting you to accept retrieving the schema from the defined SPSS file.
3. Click **Yes** to close the message and proceed to the next step.
   - A second message opens up prompting you to accept propagating the changes.
4. Click **Yes** to close the message and proceed to the next step.
5. Save the Job and press **F6** to execute it.

**Results**

The SPSS file is read row by row and the extracted fields are displayed on the log console after translating the stored values.
Starting job SPSS at 10:32 20/01/2010.

[statistics] connecting to socket on port 3418
[statistics] connected
Under 21|Female|57997.0|198522.0
21-25|Female|57113.0|203200.0
25-30|Female|54123.0|200744.0
Under 21|Male|63936.0|187791.0
21-25|Male|64335.0|195714.0
25-30|Male|66804.0|208239.0
[statistics] disconnected
Job SPSS ended at 10:32 20/01/2010. [exit code=0]
tSPSSOOutput

Writes or appends data to an SPSS .sav file. It creates SPSS files on the fly and overwrites existing ones.

tSPSSOOutput writes data entries in an .sav file.

**tSPSSOOutput Standard properties**

These properties are used to configure tSPSSOOutput running in the Standard Job framework.

The Standard tSPSSOOutput component belongs to the Business Intelligence family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Sync schema</th>
<th>Click this button to synchronize with the columns of the SPSS .sav file.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields. Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the <strong>[Repository Content]</strong> window.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: The schema is created and stored locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Filename</strong></td>
<td>Name or path of the SPSS .sav file to be written.</td>
</tr>
<tr>
<td><strong>Write Type</strong></td>
<td>Select an operation from the list:</td>
</tr>
<tr>
<td></td>
<td><strong>Write</strong>: simply writes the new data.</td>
</tr>
</tbody>
</table>
Append: writes the new data at the end of the existing data.

Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

Global Variables

| Global Variables | NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component can not be used as start component. It requires an input flow. Warning: Before being able to benefit from all functional objectives of the SPSS components, make sure to do the following: -If you have already installed SPSS, add the path to the SPSS directory as the following: SET PATH=%PATH%;<DR>:\program\SPSS, or -If you have not installed SPSS, you must copy the SPSS IO "spssio32.dll" lib from the SPSS installation CD and paste it in Talend root directory. |

| Limitation | Oracle provides two kinds of JVM platforms (32-bit and 64-bit). By default, the JVM used in a 64-bit operating system is the 64-bit version of the JVM. Since the JSPSS.dll file used by this component is compiled in the 32-bit JVM, it is needed to configure the 32-bit JVM for Job execution in Talend Studio installed on a 64-bit operating system. To do so, perform the following: 1. Open the relevant Job in the workspace. |
Scenario: Writing data in an .sav file

This Java scenario describes a very simple Job that writes data entries in an .sav file.

Setting up the Job

Procedure

1. Drop a tRowGenerator component and a tSPSSOutput component from the Palette onto the design workspace.
2. Right-click on tRowGenerator and connect it to tSPSSOutput using a Main Row link.

Configuring the input component

Procedure

1. In the design workspace, double click tRowGenerator to display its Basic Settings view and open its editor. Here you can define your schema.
2. Click the plus button to add the columns you want to write in the .sav file.
3. Define the schema and set the parameters to the columns.

⚠️ **Warning:**

*Make sure to define the length of your columns. Otherwise, an error message will display when building your Job.*

4. Click **OK** to validate your schema and close the editor.

**Configuring the output component**

**Procedure**

1. Click **tSPSSOutput** to display its **Basic settings** view and define the component properties.

2. Click the three-dot button next to the **Filename** field and browse to the SPSS .sav file in which you want to write data.

3. Click the three-dot button next to **Sync columns** to synchronize columns with the previous component. In this example, the schema to be inserted in the .sav file consists of the two columns: *id* and *country*.

4. If required, click **Edit schema** to view/edit the defined schema.

5. From the **Write Type** list, select **Write** or **Append** to simply write the input data in the .sav file or add it to the end of the .sav file.
Job execution

Save the Job and press **F6** to execute it.

The data generated by the `tRowGenerator` component is written in the defined .sav file.
tSPSSProperties

Obtains information about the main properties of a defined SPSS .sav file.
tSPSSProperties describes the properties of a defined SPSS .sav file.

tSPSSProperties Standard properties

These properties are used to configure tSPSSProperties running in the Standard Job framework.
The Standard tSPSSProperties component belongs to the Business Intelligence family.
The component in this framework is generally available.

Basic settings

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
The schema metadata in this component is predefined and thus read-only. You can click Edit schema to view the predefined metadata. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Filename</td>
<td>Name or path of the .sav file to be processed.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

Global Variables

| Global Variables | IS_VALID_FILE: the result of the file validation. This is an After variable and it returns a boolean. ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

Use this component as a start component. It needs an output flow.

⚠️ Warning:

In order to benefit from all of the functional objectives of the SPSS components, do the following: -If you have already installed SPSS, add the path to the SPSS directory as the following: SET PATH=%PATH%;<DR>:\program \SPSS, or -If you have not installed SPSS, you must copy the SPSS IO “spssio32.dll” lib from the SPSS installation CD and paste it in the Talend root directory.

Limitation

Oracle provides two kinds of JVM platforms (32-bit and 64-bit). By default, the JVM used in a 64-bit operating system is the 64-bit version of the JVM. Since the JSPSS.dll file used by this component is compiled in the 32-bit JVM, it is needed to configure the 32-bit JVM for Job execution in Talend Studio installed on a 64-bit operating system.

To do so, perform the following:

1. Open the relevant Job in the workspace.
2. Click the Run tab and enter the Advanced settings view.
3. Select the Use specific JVM arguments check box and click the New... button.
4. In the Set the VM argument box, enter -d32 and click Ok for validation.

Related scenarios

For related topics, see:

- Reading master data from an MDM hub on page 2149.
- Scenario: Writing data in an .sav file on page 3596.
tSPSSStructure

Retrieves information about the variables inside .sav files.

tSPSSStructure addresses variables inside .sav files. You can use this component in combination with tFileList to gather information about existing *.sav files to further analyze or check the findings.

**tSPSSStructure Standard properties**

These properties are used to configure tSPSSStructure running in the Standard Job framework.

The Standard tSPSSStructure component belongs to the Business Intelligence family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.
|                       | The schema metadata in this component is predefined and thus read-only. It is based on the internal SPSS convention. You can click Edit schema to view the predefined metadata. |
|                       | **Built-in**: The schema is created and stored locally for this component only. Related topic: see Talend Studio User Guide. |
|                       | **Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide. |
| Filename               | Name or path of the .sav file to be processed. |

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Global Variables**

| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. |
|                 | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Use this component as a start component. It needs an output flow.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Warning:</strong></td>
<td>Before being able to benefit from all functional objectives of the SPSS components, make sure to do the following: - If you have already installed SPSS, add the path to the SPSS directory as the following: SET PATH=%PATH%;&lt;DR&gt;\program\SPSS, or - If you have not installed SPSS, you must copy the SPSS IO “spssio32.dll” lib from the SPSS installation CD and paste it in Talend root directory.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Limitation</th>
<th>Oracle provides two kinds of JVM platforms (32-bit and 64-bit). By default, the JVM used in a 64-bit operating system is the 64-bit version of the JVM. Since the JSPSS.dll file used by this component is compiled in the 32-bit JVM, it is needed to configure the 32-bit JVM for Job execution in Talend Studio installed on a 64-bit operating system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>To do so, perform the following:</td>
<td>- Open the relevant Job in the workspace.</td>
</tr>
<tr>
<td>1.</td>
<td>Click the Run tab and enter the Advanced settings view.</td>
</tr>
<tr>
<td>2.</td>
<td>Select the Use specific JVM arguments check box and click the New... button.</td>
</tr>
<tr>
<td>3.</td>
<td>In the Set the VM argument box, enter -d32 and click Ok for validation.</td>
</tr>
</tbody>
</table>

**Related scenarios**

For related topics, see:

- Reading master data from an MDM hub on page 2149.
- Scenario: Writing data in an .sav file on page 3596.
tSQLDWHBulkExec

Loads data into an Azure SQL Data Warehouse table from either Azure Blob Storage or Azure Data Lake Store.

For more information about loading data into Azure SQL Data Warehouse, see Designing Extract, Load, and Transform (ELT) for Azure SQL Data Warehouse.

tSQLDWHBulkExec Standard properties

These properties are used to configure tSQLDWHBulkExec running in the Standard Job framework. The Standard tSQLDWHBulkExec component belongs to two families: Cloud and Databases. The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to: 1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection. 2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>JDBC Provider</td>
<td>Select the provider of the JDBC driver to be used.</td>
</tr>
<tr>
<td>Host</td>
<td>Specify the IP address or hostname of the Azure SQL Data Warehouse to be used.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Specify the listening port number of the Azure SQL Data Warehouse to be used.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Enter the name of the Azure SQL Data Warehouse schema.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Specify the name of the Azure SQL Data Warehouse to be used.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Enter the user authentication data to access the Azure SQL Data Warehouse. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Additional JDBC Parameters</strong></td>
<td>Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair. For example, <code>encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;</code> for Azure SQL database connection.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Specify the name of the SQL Data Warehouse table into which data will be loaded.</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td>Select an operation to be performed on the table defined.</td>
</tr>
<tr>
<td></td>
<td>- <strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Drop and create table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Create table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Create table if not exists</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Drop table if exists and create</strong>: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Clear table</strong>: The table content is deleted. You have the possibility to rollback the operation.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Truncate table</strong>: The table content is deleted. You do not have the possibility to rollback the operation.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>
Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Azure Storage</th>
<th>Select the type of the Azure Storage from which data will be loaded, either <strong>Blob Storage</strong> or <strong>Data Lake Store</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account Name</td>
<td>Enter the account name for your Azure Blob Storage or Azure Data Lake Store to be accessed.</td>
</tr>
<tr>
<td>Access key</td>
<td>Enter the key associated with the storage account you need to access. Two keys are available for each account and by default, either of them can be used for this access. This property is available when <strong>Blob Storage</strong> is selected from the <strong>Azure Storage</strong> drop-down list.</td>
</tr>
<tr>
<td>Container</td>
<td>Enter the name of the blob container. This property is available when <strong>Blob Storage</strong> is selected from the <strong>Azure Storage</strong> drop-down list.</td>
</tr>
<tr>
<td>Authentication key</td>
<td>Enter the authentication key needed to access your Azure Data Lake Store. This property is available when <strong>Data Lake Store</strong> is selected from the <strong>Azure Storage</strong> drop-down list.</td>
</tr>
<tr>
<td>Client Id</td>
<td>Enter your application ID (also called client ID). This property is available when <strong>Data Lake Store</strong> is selected from the <strong>Azure Storage</strong> drop-down list.</td>
</tr>
<tr>
<td>OAuth 2.0 token endpoint</td>
<td>Copy-paste the OAuth 2.0 token endpoint that you can obtain from the Endpoints list accessible on the App registrations page. This property is available when <strong>Data Lake Store</strong> is selected from the <strong>Azure Storage</strong> drop-down list.</td>
</tr>
<tr>
<td>Azure Storage Location</td>
<td>Specify the location where your Azure Blob Storage or Azure Data Lake Store account is created.</td>
</tr>
</tbody>
</table>
### Advanced settings

<table>
<thead>
<tr>
<th><strong>File format</strong></th>
<th>Select the file format that defines external data stored in your Azure Blob Storage or Azure Data Lake Store, Delimited Text, Hive RCFile, Hive ORC, or Parquet. For more information about the file formats, see CREATE EXTERNAL FILE FORMAT.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Field separator</strong></td>
<td>Specify the character(s) that indicate the end of each field in the delimited text file. This property is available when Delimited Text is selected from the File format drop-down list.</td>
</tr>
<tr>
<td><strong>Enclosed by</strong></td>
<td>Select this check box and in the field next to it, specify the character that encloses the string in the delimited file. This property is available when Delimited Text is selected from the File format drop-down list.</td>
</tr>
<tr>
<td><strong>Date format</strong></td>
<td>Select this check box and in the field next to it, specify the custom format for all date and time data in the delimited file. For more information about the date format, see CREATE EXTERNAL FILE FORMAT. This property is available when Delimited Text is selected from the File format drop-down list.</td>
</tr>
<tr>
<td><strong>Use type default</strong></td>
<td>Select this check box to store each missing value using the default value of the data type of the corresponding column. Clear this check box to store each missing value in the delimited file as NULL. This property is available when Delimited Text is selected from the File format drop-down list.</td>
</tr>
<tr>
<td><strong>Serde Method</strong></td>
<td>Select a Hive serializer and deserializer method. This property is available when Hive RCFile is selected from the File format drop-down list.</td>
</tr>
<tr>
<td><strong>Compressed by</strong></td>
<td>Select this check box if external data is compressed, and from the drop-down list displayed next to it, select the compression method.</td>
</tr>
</tbody>
</table>
| **Data import reject options** | Select this check box to specify the following reject options.  
  - **Reject type**: Specify how you want to deal with reject rows.  
  - **Value**: If the number of rejected rows exceeds the value specified in the Reject value field, the load fails.  
  - **Percentage**: If the percentage of rejected rows exceeds the value specified in the Reject value field, the load fails. |
- **Reject value**: The reject value according to the reject type. For percentage, it is the percent value without the symbol %.
- **Reject sample value**: The reject percentage sample value.

For more information about the reject options, see CREATE EXTERNAL TABLE.

### Distribution Option

Select the sharding pattern used to distribute data in the table, Round Robin, Hash, or Replicate. For more information about the sharding pattern supported by Azure SQL Data Warehouse, see Azure SQL Data Warehouse - Massively parallel processing (MPP) architecture.

This property is available when any option related to table creation is selected from the Action on table drop-down list.

### Distribution Column Name

The name of the distribution column for a hash-distribution table.

This property is available when Hash is selected from the Distribution Option drop-down list.

### Table Option

Select the index type of the table, Clustered Columnstore Index, Heap, or Clustered Index. For more information, see Indexing tables in SQL Data Warehouse.

This property is available when any option related to table creation is selected from the Action on table drop-down list.

### Index column(s)

Specify the name of one or more key columns in the index. If multiple columns are specified, separate them with comma.

This property is available when Clustered Index is selected from the Table Option drop-down list.

### Partition

Select this check box to specify the following partition options:

- **Partition column name**: Specify the name of the column used to partition the table.
- **Range**: Specify how the limit value is included in the range of the limit.
  - **Left**: The limit value is included in the left range of the limit.
  - **Right**: The limit value is included in the right range of the limit.
- **Partition For Values**: Specify the values (separated by comma) used for partition.

For more information about the table partition, see Partitioning tables in SQL Data Warehouse.
tSQLDWHBulkExec

This property is available when any option related to table creation is selected from the **Action on table** drop-down list.

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

### Global Variables

<table>
<thead>
<tr>
<th><strong>ERROR_MESSAGE</strong></th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE_INSERTED</strong></td>
<td>The number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as a standalone component of a Job or Subjob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Note that some features that are supported by other databases are not supported by Azure SQL Data Warehouse. For more information, see <strong>Unsupported table features</strong>.</td>
</tr>
</tbody>
</table>

### Related scenario

No scenario is available for this component yet.
**tSQLDWHClose**

Closes an active connection to an Azure SQL Data Warehouse database.

**tSQLDWHClose Standard properties**

These properties are used to configure tSQLDWHClose running in the Standard Job framework. The Standard tSQLDWHClose component belongs to two families: Cloud and Databases. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Component List</th>
<th>Select the tSQLDWHConnection component that opens the connection you need to close from the list.</th>
</tr>
</thead>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
</table>

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with other Azure SQL Data Warehouse components, especially with the tSQLDWHConnection and tSQLDWHCommit components.</th>
</tr>
</thead>
</table>

**Dynamic settings**

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using...
| Limitation | Note that some features that are supported by other databases are not supported by Azure SQL Data Warehouse. For more information, see [Unsupported table features](#). |

**Related scenario**

No scenario is available for this component yet.
tSQLDWHCommit

Commits in one go a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

If you need to commit each statement as an individual transaction, you have to use the **Auto Commit** function available in the connection component.

**tSQLDWHCommit Standard properties**

These properties are used to configure tSQLDWHCommit running in the Standard Job framework.

The Standard tSQLDWHCommit component belongs to two families: Cloud and Databases.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Component List</th>
<th>Select the tSQLDWHConnection component in the list if more than one connection are planned for the current Job.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task. Note that if you want to use a Row &gt; Main connection to link tSQLDWHCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global Variables**

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

**Usage**

| Usage rule | This component is more commonly used with other Azure SQL Data Warehouse components, especially with the tSQLDWHConnection and tSQLDWHRollback components. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |
| Limitation | Note that some features that are supported by other databases are not supported by Azure SQL Data Warehouse. For more information, see Unsupported table features. |

### Related scenario

No scenario is available for this component yet.
tSQLDWHConnection

Opens a connection to an Azure SQL Data Warehouse database.

**tSQLDWHConnection Standard properties**

These properties are used to configure tSQLDWHConnection running in the Standard Job framework. The Standard tSQLDWHConnection component belongs to two families: Cloud and Databases. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td><strong>JDBC Provider</strong></td>
<td>Select the provider of the JDBC driver to be used.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Specify the IP address or hostname of the Azure SQL Data Warehouse to be used.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Specify the listening port number of the Azure SQL Data Warehouse to be used.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Enter the name of the Azure SQL Data Warehouse schema.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Specify the name of the Azure SQL Data Warehouse to be used.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Enter the user authentication data to access the Azure SQL Data Warehouse. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Additional JDBC Parameters</strong></td>
<td>Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair. For example, encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30; for Azure SQL database connection.</td>
</tr>
</tbody>
</table>
### Use or register a shared DB Connection
Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the **Shared DB Connection Name** field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the `tRunJob` component. Using a shared connection together with a `tRunJob` component with either of these two options enabled will cause your Job to fail.

This check box is not available when the **Specify a data source alias** check box is selected.

### Specify a data source alias
Select this check box and in the **Data source alias** field displayed, specify the alias of a data source created on Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see *Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database* on page 2503.

This check box is not available when the **Use or register a shared DB Connection** check box is selected.

### Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Auto Commit</strong></td>
<td>Select this check box to commit any changes to the database automatically upon the transaction.</td>
</tr>
<tr>
<td></td>
<td>With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component. Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.</td>
</tr>
<tr>
<td><strong>Share identity insert in multi table</strong></td>
<td>Select this check box to share IDENTITY_INSERT with multiple tables in one connection.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>
Global Variables

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

Usage

| Usage rule | This component is more commonly used with other Azure SQL Data Warehouse components and it opens a connection that can be reused by them. |
| Limitation | Note that some features that are supported by other databases are not supported by Azure SQL Data Warehouse. For more information, see Unsupported table features. |

Related scenario

No scenario is available for this component yet.
tSQLDWHInput

Reads data and extracts fields based on a query from an Azure SQL Data Warehouse database.

**tSQLDWHInput Standard properties**

These properties are used to configure tSQLDWHInput running in the Standard Job framework.

The Standard tSQLDWHInput component belongs to two families: Cloud and Databases.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to: 1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection. 2. In the child level, use a dedicated connection component to read that registered database connection. For an example about how to share a database connection across Job levels, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set. • <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually. • <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td>JDBC Provider</td>
<td>Select the provider of the JDBC driver to be used.</td>
</tr>
<tr>
<td>Host</td>
<td>Specify the IP address or hostname of the Azure SQL Data Warehouse to be used.</td>
</tr>
<tr>
<td>Port</td>
<td>Specify the listening port number of the Azure SQL Data Warehouse to be used.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Enter the name of the Azure SQL Data Warehouse schema.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Specify the name of the Azure SQL Data Warehouse to be used.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Enter the user authentication data to access the Azure SQL Data Warehouse. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.  
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.  
Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.  
- **View schema**: choose this option to view the schema only.  
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.  
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window. |
| **Table name** | Specify the name of the table to be used. |
| **Query Type** | Select the way the query will be set.  
- **Built-in**: Fill in manually the query statement or build it graphically using SQLBuilder.  
- **Repository**: Select the relevant query stored in the Repository. The Query field gets accordingly filled in. |
<p>| <strong>Guess Query</strong> | Click this button to generate the query which corresponds to your table schema in the Query field. |
| <strong>Guess schema</strong> | Click this button to generate the schema columns based on the query specified in the Query field. |</p>
<table>
<thead>
<tr>
<th><strong>Query</strong></th>
<th>Specify the query paying particularly attention to properly sequence the fields in order to match the schema definition.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Specify a data source alias</strong></td>
<td>Select this check box and in the Data source alias field displayed, specify the alias of a data source created on Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503. This check box is not available when the <strong>Use an existing Connection</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Advanced settings</strong></td>
<td>Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair. For example, <code>encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;</code> for Azure SQL database connection.</td>
</tr>
<tr>
<td><strong>Trim all the String/Char columns</strong></td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</td>
</tr>
<tr>
<td><strong>Trim column</strong></td>
<td>Select the check box(es) in the Trim column to remove leading and trailing whitespace from the corresponding column(s). This property is not available if the <strong>Trim all the String/Char columns</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
<tr>
<td><strong>Global Variables</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td><strong>NB_LINE</strong></td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>QUERY</strong></td>
<td>The query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td><strong>Usage</strong></td>
<td>This component is usually used as a start component of a Job or Subjob and it always needs an output link.</td>
</tr>
</tbody>
</table>
**Dynamic settings**

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Limitation**

Note that some features that are supported by other databases are not supported by Azure SQL Data Warehouse. For more information, see Unsupported table features.

### Related scenario

No scenario is available for this component yet.
tSQLDWHOutput

Writes, updates, makes changes or suppresses entries in an Azure SQL Data Warehouse database.

**tSQLDWHOutput Standard properties**

These properties are used to configure tSQLDWHOutput running in the Standard Job framework. The Standard tSQLDWHOutput component belongs to two families: Cloud and Databases. The component in this framework is generally available.

**Basic settings**

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:
|**1.** In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
|**2.** In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see Talend Studio User Guide. |
| Property Type | Select the way the connection details will be set.
|**Built-In:** The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.
|**Repository:** The connection details stored centrally in Repository > Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in. |
| JDBC Provider | Select the provider of the JDBC driver to be used. |
| Host | Specify the IP address or hostname of the Azure SQL Data Warehouse to be used. |
| Port | Specify the listening port number of the Azure SQL Data Warehouse to be used. |
| **Schema** | Enter the name of the Azure SQL Data Warehouse schema. |
| **Database** | Specify the name of the Azure SQL Data Warehouse to be used. |
| **Username and Password** | Enter the user authentication data to access the Azure SQL Data Warehouse. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Table** | Specify the name of the table to be used. |
| **Action on table** | Select an operation to be performed on the table defined.  
- **Default**: No operation is carried out.  
- **Drop and create table**: The table is removed and created again.  
- **Create table**: The table does not exist and gets created.  
- **Create table if does not exist**: The table is created if it does not exist.  
- **Drop table if exist and create**: The table is removed if it already exists and created again.  
- **Clear table**: The table content is deleted. You have the possibility to rollback the operation.  
- **Truncate**: The table content is deleted. |
| **Turn on identity insert** | Select this check box to use your own sequence for the identity value of the inserted records (instead of having the SQL Server pick the next sequential value). |
| **Action on data** | Select an action to be performed on data of the table defined.  
- **Insert**: Add new entries to the table. If duplicates are found, job stops.  
- **Single Insert Query**: Add entries to the table in a batch.  
- **Update**: Make changes to existing entries.  
- **Insert or update**: Insert a new record. If the record with the given reference already exists, an update would be made.  
- **Update or insert**: Update the record with the given reference. If the record does not exist in the index pool, a new record would be inserted.  
- **Delete**: Remove entries corresponding to the input flow.  
- **Insert if not exist**: Add new entries to the table if they do not exist. |
It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking Edit Schema and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the Advanced settings view where you can simultaneously define primary keys for the Update and Delete operations. To do that, select the Use field options check box and then in the Key in update column, select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the Key in delete column for the Delete operation.

The dynamic schema feature can be used in the following modes: Insert, Update, Insert or update, Update or insert, and Delete.

### Specify identity field

Select this check box to specify the identity field, which is made up of an automatically incrementing identification number.

When this check box is selected, three other fields are displayed:

- **Identity field**: select the column you want to define as the identity field from the list.
- **Start value**: type in a start value, used for the very first row loaded into the table.
- **Step**: type in an incremental value, added to the value of the previous row that was loaded.

This check box is available only when you select Drop and create table, Create table, Create table if not exists, or Drop table if exists and create from the Action on table list, and will disappear if you select the Enable parallel execution check box in the Advanced settings view. If you select this check box with the Turn on identity insert check box cleared and the Create table if not exists option selected from the Action on table list and if the specified table does not exist, only a table will be created without inserting data into it.

Note that you can also specify the identity field from the schema of the component. To do so, set the DB Type of the relevant column to INT IDENTITY.

When the Specify identity field check box is selected, the DB Type INT IDENTITY in the schema is ignored.

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in
various projects and Job designs. Related topic: see *Talend Studio User Guide*.

Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to *Built-in* for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the *Repository Content* window.

### Specify a data source alias
Select this check box and in the **Data source alias** field displayed, specify the alias of a data source created on Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see *Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database* on page 2503.

This check box is not available when the **Use an existing Connection** check box is selected.

### Die on error
Select the check box to stop the execution of the Job when an error occurs.

Clear the check box to skip any rows on error and complete the process for error-free rows.

### Advanced settings

#### Additional JDBC Parameters
Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair. For example, `encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;` for Azure SQL database connection.

#### Commit every
Enter the number of rows to be completed before committing batches of rows together into the database. This option ensures transaction quality (but not rollback) and above all better performance on executions.

#### Additional Columns
This option is not offered if you create (with or without drop) the database table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.

- **Name**: Type in the name of the schema column to be altered or inserted as new column.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SQL expression</strong></td>
<td>Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td>Select Before, Replace or After following the action to be performed on the reference column.</td>
</tr>
<tr>
<td><strong>Reference column</strong></td>
<td>Type in a column of reference that can be used to place or replace the new or altered column.</td>
</tr>
<tr>
<td><strong>Use field options</strong></td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td><strong>Ignore date out of range</strong></td>
<td>Select this check box to ignore the date validation and insert the data directly into the database for the data types of DATE, DATETIME, DATETIME2 and DATETIMEOFFSET.</td>
</tr>
<tr>
<td><strong>Enable debug mode</strong></td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td><strong>Support null in “SQL WHERE” statement</strong></td>
<td>Select this check box if you want to deal with the Null values contained in a database table. Make sure that the Nullable check box is selected for the corresponding columns in the schema.</td>
</tr>
<tr>
<td><strong>Use Batch</strong></td>
<td>Select this check box to activate the batch mode for data processing. This check box is available only when you have selected the Insert, the Update, the Single Insert Query or the Delete option in the Action on data list.</td>
</tr>
<tr>
<td><strong>Batch Size</strong></td>
<td>Specify the number of records to be processed in each batch. This field appears only when the Use batch mode check box is selected.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
<tr>
<td>NB_LINE</td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>NB_LINE_UPDATED</td>
<td>The number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_INSERTED</td>
<td>The number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_DELETED</td>
<td>The number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_REJECTED</td>
<td>The number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>

**Usage**

**Usage rule**

This component is usually used as an end component of a Job or Subjob and it always needs an input link.

**Dynamic settings**

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

**Limitation**

When the Single Insert Query option is selected in the Action on data list, an SQL Prepared Statement is generated, for example, `INSERT INTO table (col1, col2, col3) VALUES (?, ?, ?), (?, ?, ?), (?, ?, ?), (?, ?, ?)`. Within brackets are the groups of parameters the number of which cannot exceed 2000, generally, depending on the JDBC driver. Therefore, the batch size should be set so that this limit is respected.

Note that some features that are supported by other databases are not supported by Azure SQL Data Warehouse. For more information, see Unsupported table features.

Also note that when creating or deleting a table with this component, it is recommended to use the auto commit function by reusing the database connection.
Related scenario

No scenario is available for this component yet.
**tSQLDWHRollback**

Cancels the transaction commit in the connected Azure SQL Data Warehouse database to prevent partial transaction commit if an error occurs.

**tSQLDWHRollback Standard properties**

These properties are used to configure tSQLDWHRollback running in the Standard Job framework. The Standard tSQLDWHRollback component belongs to two families: Cloud and Databases. The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Component List</th>
<th>Select the <strong>tSQLDWHConnection</strong> component in the list if more than one connection are planned for the current Job.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the operation is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

### Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

### Global Variables

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

### Usage

| Usage rule | This component is more commonly used with other Azure SQL Data Warehouse components, especially with the **tSQLDWHConnection** and **tSQLDWHRollback** components. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, |
when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

| Limitation | Note that some features that are supported by other databases are not supported by Azure SQL Data Warehouse. For more information, see Unsupported table features. |

**Related scenario**

No scenario is available for this component yet.
tSQLDWHRow

Executes an SQL query stated on an Azure SQL Data Warehouse database.

**tSQLDWHRow Standard properties**

These properties are used to configure tSQLDWHRow running in the Standard Job framework.

The Standard tSQLDWHRow component belongs to two families: Cloud and Databases.

The component in this framework is generally available.

**Basic settings**

| Use an existing connection | Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide. |
| --- | --- |
| Property Type | Select the way the connection details will be set.

- **Built-In:** The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.

- **Repository:** The connection details stored centrally in Repository > Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in. |
| JDBC Provider | Select the provider of the JDBC driver to be used. |
| Host | Specify the IP address or hostname of the Azure SQL Data Warehouse to be used. |
| Port | Specify the listening port number of the Azure SQL Data Warehouse to be used. |
| **Schema** | Enter the name of the Azure SQL Data Warehouse schema. |
| **Database** | Specify the name of the Azure SQL Data Warehouse to be used. |
| **Username and Password** | Enter the user authentication data to access the Azure SQL Data Warehouse. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields.  
  - **Built-In**: You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](#).  
  - **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](#).  
Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. |
| **Table name** | Specify the name of the table to be used. |
| **Turn on identity insert** | Select this check box to use your own sequence for the identity value of the inserted records (instead of having the SQL Server pick the next sequential value). |
| **Query Type** | Select the way the query will be set.  
  - **Built-in**: Fill in manually the query statement or build it graphically using SQLBuilder.  
  - **Repository**: Select the relevant query stored in the Repository. The **Query** field gets accordingly filled in. |
<table>
<thead>
<tr>
<th><strong>Guess Query</strong></th>
<th>Click the <strong>Guess Query</strong> button to generate the query which corresponds to your table schema in the <strong>Query</strong> field.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query</strong></td>
<td>Specify your database query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Specify a data source alias</strong></td>
<td>Select this check box and in the <strong>Data source alias</strong> field displayed, specify the alias of a data source created on Talend Runtime side to use the shared connection pool defined in the data source configuration. This option works only when you deploy and run your Job in Talend Runtime. For a related use case, see Scenario: Deploying your Job on Talend Runtime to retrieve data from a MySQL database on page 2503. This check box is not available when the <strong>Use an existing Connection</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a <strong>Row &gt; Reject</strong> connection.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Advanced settings</strong></th>
<th>Specify additional connection properties for the database connection you are creating. The properties are separated by semicolon and each property is a key-value pair. For example, <code>encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;</code> for Azure SQL database connection.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional JDBC Parameters</strong></td>
<td>Select this check box to insert the result of the query into a column of the current flow. Select this column from the <strong>use column</strong> list. This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of Object and this component is usually followed by <strong>tParseRecordSet</strong>.</td>
</tr>
<tr>
<td><strong>Propagate QUERY’s recordset</strong></td>
<td>Select this check box if you want to query the database using a PreparedStatement. In the <strong>Set PreparedStatement Parameters</strong> table, define the parameters represented by “?” in the SQL instruction of the <strong>Query</strong> field in the <strong>Basic Settings</strong> tab.</td>
</tr>
<tr>
<td><strong>Use PreparedStatement</strong></td>
<td><strong>Parameter Index</strong>: Enter the parameter position in the SQL instruction.</td>
</tr>
<tr>
<td></td>
<td><strong>Parameter Type</strong>: Enter the parameter type.</td>
</tr>
<tr>
<td></td>
<td><strong>Parameter Value</strong>: Enter the parameter value.</td>
</tr>
</tbody>
</table>
This option is very useful if you need to execute the same query several times. Performance levels are increased.

<table>
<thead>
<tr>
<th>Commit every</th>
<th>Enter the number of rows to be completed before committing batches of rows together into the database. This option ensures transaction quality (but not rollback) and above all better performance on executions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>ERROR_MESSAGE</th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUERY</td>
<td>The query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+ button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Limitation

| Limitation | Note that some features that are supported by other databases are not supported by Azure SQL Data Warehouse. For more information, see Unsupported table features. |
Also note that when creating or deleting a table with this component, it is recommended to use the auto commit function by reusing the database connection created by a `tSQLDWHConnection` component and selecting the `Auto Commit` check box on the `Advanced settings` view of the `tSQLDWHConnection` component, instead of using a `tSWLDWHCommit` component.

### Related scenario

No scenario is available for this component yet.
tSQLiteClose

Closes a transaction committed in the connected DB.

**tSQLiteClose Standard properties**

These properties are used to configure tSQLiteClose running in the Standard Job framework.

The Standard tSQLiteClose component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tSQLiteConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is to be used along with SQLite components, especially with tSQLiteConnection and tSQLiteCommit. |
| Dynamic settings | Click the [*] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context- |
Related scenarios

No scenario is available for the Standard version of this component yet.
tSQLiteCommit

Commits in one go, using a unique connection, a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

TSQLiteCommit validates the data processed through the Job into the connected DB

TSQLiteCommit Standard properties

These properties are used to configure tSQLiteCommit running in the Standard Job framework.

The Standard tSQLiteCommit component belongs to the Databases family.

The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tSQLiteConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task. Warning: If you want to use a Row &gt; Main connection to link tSQLiteCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

Usage

| Usage rule | This component is more commonly used with other tSQLite* components, especially with the tSQLiteConnection and tSQLiteRollback components. |
Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Related scenario**

For **tSQLiteCommit** related scenario, see **Scenario: Inserting data in mother/daughter tables** on page 2476.
tSQLiteConnection

Opens a connection to the database for a current transaction.

**tSQLiteConnection Standard properties**

These properties are used to configure tSQLiteConnection running in the Standard Job framework.

The Standard tSQLiteConnection component belongs to the Databases and the ELT families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
</tbody>
</table>
| **Use or register a shared DB Connection** | Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the **Shared DB Connection Name** field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the tRunJob component. Using a shared connection together with a tRunJob component with either of these two options enabled will cause your Job to fail.

**Advanced settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Auto Commit</strong></td>
<td>Select this check box to commit any changes to the database automatically upon the transaction. With this check box selected, you cannot use the corresponding commit component to commit changes</td>
</tr>
</tbody>
</table>
to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

| tStatCatcher Statistics | Select this check box to gather the job processing metadata at a Job level as well as at each component level. |

**Usage**

| Usage rule | This component is more commonly used with other tSQLite* components, especially with the tSQLiteCommit and tSQLiteRollback components. |

**Related scenarios**

For **tSQLiteConnection** related scenario, see **tMysqlConnection** on page 2475.
tSQLiteInput

Executes a DB query with a defined command which must correspond to the schema definition. It passes on rows to the next component via a Main row link.
tSQLiteInput reads a database file and extracts fields based on an SQL query. As it embeds the SQLite engine, no need of connecting to any database server.

**tSQLiteInput Standard properties**

These properties are used to configure tSQLiteInput running in the Standard Job framework.
The Standard tSQLiteInput component belongs to the Databases family.
The component in this framework is generally available.

*i Note:*

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
</tbody>
</table>

- **Built-in:** No property data stored centrally.
- **Repository:** Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
</table>

*i Note:*

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Filepath to the SQLite database file.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <em>line</em> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>The schema is created and stored locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>The schema already exists and is stored in the Repository, hence can be reused. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the *[Repository Content]* window.

<table>
<thead>
<tr>
<th><strong>Query type</strong></th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query</strong></td>
<td>If your query is not stored in the Repository, type in your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Trim all the String/Char columns</strong></th>
<th>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trim column</strong></td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

### Global Variables

| Global Variables | **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is standalone as it includes the SQLite engine. This is a startable component that can initiate a data flow processing.</th>
</tr>
</thead>
</table>

| Dynamic settings | Click the [+ ] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  
The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.  
For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------|
Scenario: Filtering SQLite data

This scenario describes a rather simple job which uses a select statement based on a filter to extract rows from a source SQLite Database and feed an output SQLite table.

- Drop from the Palette, a tSQLiteInput and a tSQLiteOutput component from the Palette to the design workspace.
- Connect the input to the output using a row main link.
- On the tSQLiteInput Basic settings, type in or browse to the SQLite Database input file.

<table>
<thead>
<tr>
<th>id</th>
<th>version</th>
<th>download_date</th>
<th>ip</th>
<th>type</th>
<th>type_os</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20027</td>
<td>12/11/2006</td>
<td>1191947307</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>20028</td>
<td>13/11/2006</td>
<td>1195650472</td>
<td>1</td>
<td>102</td>
</tr>
<tr>
<td>3</td>
<td>20030</td>
<td>12/11/2006</td>
<td>3375565745</td>
<td>1</td>
<td>103</td>
</tr>
<tr>
<td>4</td>
<td>20031</td>
<td>12/11/2006</td>
<td>1195650472</td>
<td>1</td>
<td>104</td>
</tr>
<tr>
<td>5</td>
<td>20032</td>
<td>12/11/2006</td>
<td>1195650472</td>
<td>1</td>
<td>105</td>
</tr>
<tr>
<td>6</td>
<td>20033</td>
<td>12/11/2006</td>
<td>1104072453</td>
<td>1</td>
<td>106</td>
</tr>
<tr>
<td>7</td>
<td>20034</td>
<td>12/11/2006</td>
<td>1104872453</td>
<td>1</td>
<td>107</td>
</tr>
<tr>
<td>8</td>
<td>20036</td>
<td>12/11/2006</td>
<td>1190898057</td>
<td>1</td>
<td>108</td>
</tr>
<tr>
<td>9</td>
<td>20037</td>
<td>12/11/2006</td>
<td>1190898057</td>
<td>1</td>
<td>109</td>
</tr>
<tr>
<td>10</td>
<td>20038</td>
<td>12/11/2006</td>
<td>1349977142</td>
<td>1</td>
<td>110</td>
</tr>
<tr>
<td>11</td>
<td>20040</td>
<td>13/11/2006</td>
<td>3591349521</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>20041</td>
<td>13/11/2006</td>
<td>1190898057</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>20043</td>
<td>12/11/2006</td>
<td>1196485544</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>20044</td>
<td>13/11/2006</td>
<td>1066743463</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>20045</td>
<td>12/11/2006</td>
<td>1196485544</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>20046</td>
<td>12/11/2006</td>
<td>2694272841</td>
<td>1</td>
<td>16</td>
</tr>
</tbody>
</table>

- The file contains hundreds of lines and includes an ip column which the select statement will based on
- On the tSQLite Basic settings, edit the schema for it to match the table structure.

- In the Query field, type in your select statement based on the ip column.
- On the tSQLiteOutput component Basic settings panel, select the Database filepath.
• Type in the **Table** to be fed with the selected data.

• Select the **Action on table** and **Action on Data**. In this use case, the action on table is *Drop and create* and the action on data is *Insert*.

• The schema should be synchronized with the input schema.

• Save the job and run it.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>version</td>
<td>download_date</td>
<td>ip</td>
<td>type</td>
<td>type_os</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>20028</td>
<td>13/11/2006</td>
<td>1195650472</td>
<td>1</td>
<td>102</td>
</tr>
<tr>
<td>2</td>
<td>20031</td>
<td>13/11/2006</td>
<td>1195650472</td>
<td>1</td>
<td>104</td>
</tr>
<tr>
<td>3</td>
<td>20032</td>
<td>13/11/2006</td>
<td>1195650472</td>
<td>1</td>
<td>105</td>
</tr>
</tbody>
</table>

The data queried is returned in the defined SQLite file.
tSQLiteOutput

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the job.

tSQLiteOutput writes, updates, makes changes or suppresses entries in an SQLite database. As it embeds the SQLite engine, no need of connecting to any database server.

**tSQLiteOutput Standard properties**

These properties are used to configure tSQLiteOutput running in the Standard Job framework.

The Standard tSQLiteOutput component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <a href="#">Apply</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <a href="#">Built-in</a> or <a href="#">Repository</a>.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Database</th>
<th>Filepath to the Database file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td><strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create a table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td><strong>Create a table if not exists</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop a table if exists and create</strong>: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear a table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td><strong>Action on data</strong></td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Make changes to existing entries</td>
</tr>
<tr>
<td></td>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td><strong>Update or insert</strong>: Update the record with the given reference. If the record does not exist, a new record would be inserted.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Remove entries corresponding to the input flow.</td>
</tr>
</tbody>
</table>

**Warning:**

*It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based.* You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the Update and Delete operations. To do that: Select the **Use field options** check box and then in the **Key in update column**, select the check boxes next to the column names you want to use as a base for the
**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

**Built-In:** You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](https://help.talend.com).

**Repository:** You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](https://help.talend.com).

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Die on error**

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link.

**Advanced settings**

**Commit every**

Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.

**Additional Columns**

This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are...
not insert, nor update or delete actions, or action that require particular preprocessing.

| Name: | Type in the name of the schema column to be altered or inserted as new column |
| SQL expression: | Type in the SQL statement to be executed in order to alter or insert the relevant column data. |
| Position: | Select Before, Replace or After following the action to be performed on the reference column. |
| Reference column: | Type in a column of reference that the tDBoutput can use to place or replace the new or altered column. |

Use field options
Select this check box to customize a request, especially when there is double action on data.

Enable debug mode
Select this check box to display each step during processing entries in a database.

Use Batch
Select this check box to activate the batch mode for data processing.

**Note:**
This check box is available only when you have selected the Insert, Update, or Delete option in the Action on data option.

Batch Size
Specify the number of records to be processed in each batch.
This field appears only when the Use batch mode check box is selected.

**tStat Catcher Statistics**
Select this check box to collect log data at the component level.

---

**Global Variables**

| Global Variables | NB_LINE: the number of rows processed. This is an After variable and it returns an integer. |
| NB_LINE_UPDATED: the number of rows updated. This is an After variable and it returns an integer. |
| NB_LINE_INSERTED: the number of rows inserted. This is an After variable and it returns an integer. |
| NB_LINE_DELETED: the number of rows deleted. This is an After variable and it returns an integer. |
| NB_LINE_REJECTED: the number of rows rejected. This is an After variable and it returns an integer. |
| QUERY: the query statement processed. This is an After variable and it returns a string. |
| ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. |
variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

#### Usage rule

This component must be connected to an Input component. It allows you to carry out actions on a table or on the data of a table in an SQLite database. It also allows you to create reject flows using a **Row > Reject** link to filter erroneous data. For an example of **tSQLiteOutput** in use, see *Scenario: Retrieving data in error with a Reject link* on page 2529.

#### Dynamic settings

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide*.

### Related Scenario

For scenarios related to **tSQLiteOutput**, see *Scenario: Retrieving data in error with a Reject link* on page 2529.
tSQLiteRollback

Cancels the transaction committed in the SQLite database.

### tSQLiteRollback Standard properties

These properties are used to configure tSQLiteRollback running in the Standard Job framework.

The Standard tSQLiteRollback component belongs to the Databases family.

The component in this framework is generally available.

#### Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tSQLiteConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

### Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

### Usage

| Usage rule | This component is more commonly used with other tSQLite* components, especially with the tSQLiteConnection and tSQLiteCommit components. |
| Dynamic settings | Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. |
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

For tSQLiteRollback related scenario, see Scenario: Rollback from inserting data in mother/daughter tables on page 2480.
tSQLiteRow

Executes the defined query onto the specified database and uses the parameters bound with the column.

A prepared statement uses the input flow to replace the placeholders with the values for each parameters defined. This component can be very useful for updates.

**tSQLiteRow Standard properties**

These properties are used to configure tSQLiteRow running in the Standard Job framework.

The Standard tSQLiteRow component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Query type**

Either **Built-in** or **Repository**.

- **Built-in**: Fill in manually the query statement or build it graphically using SQLBuilder
- **Repository**: Select the relevant query stored in the Repository. The Query field gets accordingly filled in.

**Query**

Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.

**Die on error**

Clear this check box to skip the row on error and complete the process for error-free rows.

**Advanced settings**

**Propagate QUERY’s recordset**

Select this check box to insert the result of the query into a **COLUMN** of the current flow. Select this column from the **use column** list.

**Note:**

This option allows the component to have a different schema from that of the preceding
component. Moreover, the column that holds the QUERY’s recordset should be set to the type of Object and this component is usually followed by tParseRecordSet.

Use PreparedStatement

Select this check box if you want to query the database using a PreparedStatement. In the Set PreparedStatement Parameter table, define the parameters represented by "?" in the SQL instruction of the Query field in the Basic Settings tab.

Parameter Index: Enter the parameter position in the SQL instruction.

Parameter Type: Enter the parameter type.

Parameter Value: Enter the parameter value.

Note: This option is very useful if you need to execute the same query several times. Performance levels are increased.

Commit every

Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.

tStat Catcher Statistics

Select this check box to collect log data at the component level.

Global Variables

QUERY: the query statement being processed. This is a Flow variable and it returns a string.

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

This component offers the flexibility of the DB query and covers all possible SQL queries.

Dynamic settings

Click the [*] button to add a row in the table and fill the Code field with a context variable to choose.
your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Scenario: Updating SQLite rows

This scenario describes a job which updates an SQLite database file based on a prepared statement and using a delimited file.

- Drop a tFileInputDelimited and a tSQLiteRow component from the Palette to the design workspace.
- On the tFileInputDelimited Basic settings panel, browse to the input file that will be used to update rows in the database.
- There is no Header nor Footer. The Row separator is a carriage return and the Field separator is a semi-colon.
- Click the [...] button next to Edit schema and define the schema structure in case it is not stored in the Repository.
• Make sure the length and type are respectively correct and large enough to define the columns.

• Then in the tSQLiteRow Basic settings panel, set the Database filepath to the file to be updated.

• The schema is read-only as it is required to match the input schema.

• Type in the query or retrieve it from the Repository. In this use case, we updated the type_os for the id defined in the Input flow. The statement is as follows: "Update download set type_os=? where id=?".

• Then select the Use PreparedStatement check box to display the placeholders’ parameter table.
In the Input parameters table, add as many lines as necessary to cover all placeholders. In this scenario, `type_os` and `id` are to be defined.

Set the `Commit every` field.

Save the job and press F6 to run it.

The download table from the SQLite database is thus updated with new `type_os` code according to the delimited input file.

**Related scenarios**

For a related scenario, see:

- Scenario: Combining two flows for selective output on page 2558
tSQLTemplate

Executes the common database actions or customized SQL statement templates, for example to drop/create a table.

tSQLTemplate offers a range of SQL statement templates for a number of DBMSs to facilitate some of the most common database actions. These templates are accessible via the SQL Template view. Additionally, you are allowed to customize the SQL statement templates as needed.

tSQLTemplate Standard properties

These properties are used to configure tSQLTemplate running in the Standard Job framework.

The Standard tSQLTemplate component belongs to the ELT family.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Select the database type you want to connect to from the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the relevant database connection component in the list if you use more than one connection in the current Job.</td>
</tr>
<tr>
<td>Database name</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Table name</td>
<td>Name of the table on which the SQL statement templates will be executed.</td>
</tr>
</tbody>
</table>

Schema and Edit schema

A schema is a row description, that is to say, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job flowcharts. Related topic: see *Talend Studio User Guide*.

### Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

### Global Variables

**Global Variables**

- **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.
- **QUERY**: the query statement being processed. This is a Flow variable and it returns a string.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the *Die on error* check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press `Ctrl + Space` to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

**Usage rule**

As a start component, this component is used with other database components, especially the database connection and commit components.

**SQL Template**

**SQL Template List**

**Dynamic settings**

Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your HDFS connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access files in different HDFS systems or different distributions, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of *Talend Studio*.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defin
ed, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

For a related scenario, see Scenario: Filtering and aggregating table columns directly on the DBMS on page 3663.
tSQLTemplateAggregate

Provide a set of matrix based on values or calculations.

tSQLTemplateAggregate collects data values from one or more columns with the intent to manage the collection as a single unit. This component has real-time capabilities since it runs the data transformation on the DBMS itself.

**tSQLTemplateAggregate Standard properties**

These properties are used to configure tSQLTemplateAggregate running in the Standard Job framework.

The Standard tSQLTemplateAggregate component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Select the database type you want to connect to from the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the relevant DB connection component in the list if you use more than one connection in the current Job.</td>
</tr>
<tr>
<td>Database name</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Source table name</td>
<td>Name of the table holding the data you want to collect values from.</td>
</tr>
<tr>
<td>Target table name</td>
<td>Name of the table you want to write the collected and transformed data in.</td>
</tr>
</tbody>
</table>

**Schema and Edit schema**

A schema is a row description, that is to say, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.
### Built-in
You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

### Repository
You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job flowcharts. Related topic: see *Talend Studio User Guide*.

### Operations
Select the type of operation along with the value to use for the calculation and the output field.

#### Output Column
Select the destination field in the list.

#### Function
Select any of the following operations to perform on data: **count**, **min**, **max**, **avg**, **sum**, and **count (distinct)**.

#### Input column position
Select the input column from which you want to collect the values to be aggregated.

### Group by
Define the aggregation sets, the values of which will be used for calculations.

#### Output Column
Select the column label in the list offered according to the schema structure you defined. You can add as many output columns as you wish to make more precise aggregations.

#### Input column position
Match the input column label with your output columns, in case the output label of the aggregation set needs to be different.

### Advanced settings

#### tStatCatcher Statistics
Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

### Global Variables

#### Global Variables
**NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as an intermediate component with other relevant DB components, especially the DB connection and commit components.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Template</td>
<td>SQL Template List</td>
</tr>
</tbody>
</table>

**Scenario: Filtering and aggregating table columns directly on the DBMS**

The following scenario creates a Job that opens a connection to a Mysql database and:

- instantiates the schemas from a database table whose rows match the column names specified in the filter,
- filters a column in the same database table to have only the data that matches a WHERE clause,
- collects data grouped by specific value(s) from the filtered column and writes aggregated data in a target database table.

To filter and aggregate database table columns:

- Drop the following components from the **Palette** onto the design workspace: tELTMysqlconnection, tSQLTemplateFilterColumns, tSQLTemplateFilterRows, tSQLTemplateAggregate, tSQLTemplateCommit, and tSQLTemplateRollback.
- Connect the five first components using **OnComponentOk** links.
- Connect **tSQLTemplateAggregate** to **tSQLTemplateRollback** using an **OnComponentError** link.
- In the design workspace, select **tMysqlConnection** and click the **Component** tab to define the basic settings for **tMysqlConnection**.
• In the **Basic settings** view, set the database connection details manually or select **Repository** from the **Property Type** list and select your DB connection if it has already been defined and stored in the **Metadata** area of the **Repository** tree view.

For more information about Metadata, see *Talend Studio User Guide*.

• In the design workspace, select **tSQLTemplateFilterColumns** and click the **Component** tab to define its basic settings.

  ![tSQLTemplateFilterColumns_1](image)

  - On the **Database type** list, select the relevant database.
  - On the **Component list**, select the relevant database connection component if more than one connection is used.
  - Enter the names for the database, source table, and target table in the corresponding fields and click the three-dot buttons next to **Edit schema** to define the data structure in the source and target tables.

    **Note:**

    When you define the data structure for the source table, column names automatically appear in the **Column** list in the **Column filters** panel.

    In this scenario, the source table has five columns: `id`, `First_Name`, `Last_Name`, `Address`, and `id_State`.
  
• In the **Column filters** panel, set the column filter by selecting the check boxes of the columns you want to write in the source table.
In this scenario, the tSQLTemplateFilterColumns component instantiates only three columns: id, First_Name, and id_State from the source table.

**Note:**

In the Component view, you can click the SQL Template tab and add system SQL templates or create your own and use them within your Job to carry out the coded operation. For more information, see tSQLTemplateFilterColumns Standard properties on page 3669.

- In the design workspace, select tSQLTemplateFilterRows and click the Component tab to define its basic settings.

  ![tSQLTemplateFilterRows_1](image)

- On the Database type list, select the relevant database.

- On the Component list, select the relevant database connection component if more than one connection is used.

- Enter the names for the database, source table, and target table in the corresponding fields and click the three-dot buttons next to Edit schema to define the data structure in the source and target tables.

In this scenario, the source table has the three initially instantiated columns: id, First_Name, and id_State and the source table has the same three-column schema.

- In the Where condition field, enter a WHERE clause to extract only those records that fulfill the specified criterion.

In this scenario, the tSQLTemplateFilterRows component filters the First_Name column in the source table to extract only the first names that contain the ‘a’ letter.

- In the design workspace, select tSQLTemplateAggregate and click the Component tab to define its basic settings.

- On the Database type list, select the relevant database.

- On the Component list, select the relevant database connection component if more than one connection is used.

- Enter the names for the database, source table, and target table in the corresponding fields and click the three-dot buttons next to Edit schema to define the data structure in the source and target tables.

The schema for the source table consists of the three columns: id, First_Name, and id_State. The schema for the target table consists of two columns: customers_status and customers_number. In this scenario, we want to group customers by their marital status and count customer number in each marital group. To do that, we define the Operations and Group by panels accordingly.
• In the Operations panel, click the plus button to add one or more lines and then click in the Output column line to select the output column that will hold the counted data.

• Click in the Function line and select the operation to be carried on.

• In the Group by panel, click the plus button to add one or more lines and then click in the Output column line to select the output column that will hold the aggregated data.

• In the design workspace, select `tSQLTemplateCommit` and click the Component tab to define its basic settings.

• On the Database type list, select the relevant database.

• On the Component list, select the relevant database connection component if more than one connection is used.

• Do the same for `tSQLTemplateRollback`.

• Save your Job and press F6 to execute it.

A two-column table `aggregate_customers` is created in the database. It groups customers according to their marital status and count customer number in each marital group.

```
customers_status   customers_number
Married            1
Single             3
```
**tSQLTemplateCommit**

Commits a global action in one go using a single connection, instead of doing so for every row or every batch of rows separately. This provides a gain in performance.

tSQLTemplateCommit validates the data processed in a Job in a specified database.

**tSQLTemplateCommit Standard properties**

These properties are used to configure tSQLTemplateCommit running in the Standard Job framework.

The Standard tSQLTemplateCommit component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Select the database type you want to connect to from the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the ELT database connection component in the list if more than one connection is required for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><strong>NB_LINE</strong>: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>QUERY</strong>: the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
</tbody>
</table>
For further information about variables, see *Talend Studio User Guide*.

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with the tSQLTemplateRollback component and the ELT database connection component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Template</td>
<td>SQL Template List</td>
</tr>
</tbody>
</table>

**Related scenario**

For more information on *tSQLTemplateCommit*, see *Scenario: Filtering and aggregating table columns directly on the DBMS* on page 3663.
tSQLTemplateFilterColumns

Homogenizes schemas by reorganizing, deleting or adding new columns.

tSQLTemplateFilterColumns makes specified changes to the defined schema of the database table based on column name mapping. This component has real-time capabilities since it runs the data filtering on the DBMS itself.

**tSQLTemplateFilterColumns Standard properties**

These properties are used to configure tSQLTemplateFilterColumns running in the Standard Job framework.

The Standard tSQLTemplateFilterColumns component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Select the type of database you want to work on from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the relevant DB connection component in the list if you use more than one connection in the current Job.</td>
</tr>
<tr>
<td>Database name</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Source table name</td>
<td>Name of the table holding the data you want to filter.</td>
</tr>
<tr>
<td>Target table name</td>
<td>Name of the table you want to write the filtered data in.</td>
</tr>
</tbody>
</table>

**Schema and Edit schema**

A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.
Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job flowcharts. Related topic: see Talend Studio User Guide.

Column Filters

In the table, click the Filter check box to filter all of the columns. To select specific columns for filtering, select the check box(es) which correspond(s) to the column name(s).

Advanced settings

tStatCatcher Statistics

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

Global Variables

Global Variables

NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

QUERY: the query statement being processed. This is a Flow variable and it returns a string.

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

This component is used as an intermediary component with other relevant DB components, especially DB connection components.

SQL Template

SQL Template List

Related Scenario

For a related scenario, see Scenario: Filtering and aggregating table columns directly on the DBMS on page 3663.
tSQLTemplateFilterRows

Sets row filters for any given data source, based on a WHERE clause.
tSQLTemplateFilterRows allows you to define a row filter on one table. This component has real-time capabilities since it runs the data filtering on the DBMS itself.

**tSQLTemplateFilterRows Standard properties**

These properties are used to configure tSQLTemplateFilterRows running in the Standard Job framework.
The Standard tSQLTemplateFilterRows component belongs to the ELT family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Select the type of database you want to work on from the drop down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the relevant DB connection component in the list if you are using more than one connection in the current Job.</td>
</tr>
<tr>
<td>Database name</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Source table name</td>
<td>Name of the table holding the data you want to filter.</td>
</tr>
<tr>
<td>Target table name</td>
<td>Name of the table you want to write the filtered data in.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td>Built-in</td>
<td>You create and store the schema locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
</tbody>
</table>
Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job flowcharts. Related topic: see *Talend Studio User Guide*.

Where condition

Use a WHERE clause to set the criteria that you want the rows to meet.

You can use the WHERE clause to select specific rows from the table that match specified criteria or conditions.

**Advanced settings**

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

**Global Variables**

**Global Variables**

**NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.

**ERROR MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

**Usage rule**

This component is used as an intermediary component with other DB components, particularly DB connection components.

**SQL Template**

**SQL Template List**

**Related Scenario**

For a related scenario, see *Scenario: Filtering and aggregating table columns directly on the DBMS* on page 3663.
tSQLTemplateMerge

Merges data into a database table directly on the DBMS by creating and executing a MERGE statement.

tSQLTemplateMerge creates an SQL MERGE statement to merge data into a database table.

**tSQLTemplateMerge Standard properties**

These properties are used to configure tSQLTemplateMerge running in the Standard Job framework.

The Standard tSQLTemplateMerge component belongs to the ELT family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database Type</strong></td>
<td>Select the type of database you want to work on from the drop-down list.</td>
</tr>
<tr>
<td><strong>Component list</strong></td>
<td>Select the relevant DB connection component from the list if you use more than one connection in the current Job.</td>
</tr>
<tr>
<td><strong>Source table name</strong></td>
<td>Name of the database table holding the data you want to merge into the target table.</td>
</tr>
<tr>
<td><strong>Target table name</strong></td>
<td>Name of the table you want to merge data into.</td>
</tr>
</tbody>
</table>

**Schema and Edit schema**

This component involves two schemas: source schema and target schema.

A schema is a row description, that is to say, it defines the number of fields to be processed and passed on to the next component. The schema is either built-in or remotely stored in the Repository.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*. 
Repository: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see Talend Studio User Guide.

| **Merge ON** | Specify the target and source columns you want to use as the primary keys. |
| **Use UPDATE (WHEN MATCHED)** | Select this check box to update existing records. With the check box selected, the **UPDATE Columns** table appears, allowing you to define the columns in which records are to be updated. |
| **Specify additional output columns** | Select this check box to update records in additional columns other than those listed in the **UPDATE Columns** table. With this check box selected, the **Additional UPDATE Columns** table appears, allowing you to specify additional columns. |
| **Specify UPDATE WHERE clause** | Select this check box and type in a WHERE clause in the **WHERE clause** field to filter data during the update operation.  

**Note:** This option may not work with certain database versions, including Oracle 9i. |
| **Use INSERT (WHEN MATCHED)** | Select this check box to insert new records. With the check box selected, the **INSERT Columns** table appears, allowing you to specify the columns to be involved in the insert operation. |
| **Specify additional output columns** | Select this check box to insert records to additional columns other than those listed in the **INSERT Columns** table. With this check box selected, the **Additional INSERT Columns** table appears, allowing you to specify additional columns. |
| **Specify INSERT WHERE clause** | Select this check box and type in a WHERE clause in the **WHERE clause** field to filter data during the insert operation.  

**Note:** This option may not work with certain database versions, including Oracle 9i. |

### Advanced settings

**tStatCatcher Statistics**  
Select this check box to gather the Job processing metadata at a Job level as well as at component level.

### Global Variables

**Global Variables**  
**NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.
### tSQLTemplateMerge

**NB_LINE_MERGED**: the number of rows merged. This is an After variable and it returns an integer.

**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SQL Template</strong></td>
<td>This component is used as an intermediate component with other relevant DB components, especially the DB connection and commit components.</td>
</tr>
</tbody>
</table>

### Scenario: Merging data directly on the DBMS

This scenario describes a simple Job that opens a connection to a MySQL database, merges data from a source table into a target table according to customer IDs, and displays the contents of the target table before and after the merge action. A WHERE clause is used to filter data during the merge operation.
• Drop a tMysqlConnection component, a tSQLTemplateMerge component, two tMysqlInput components and two tLogRow components from the Palette onto the design workspace.

• Connect the tMysqlConnection component to the first tMysqlInput component using a Trigger > OnSubjobOK connection.

• Connect the first tMysqlInput component to the first tLogRow component using a Row > Main connection. This row will display the initial contents of the target table on the console.

• Connect the first tMysqlInput component to the tSQLTemplateMerge component, and the tSQLTemplateMerge component to the second tMysqlInput component using Trigger > OnSubjobOK connections.

• Connect the second tMysqlInput component to the second tLogRow component using a Row > Main connection. This row will display the merge result on the console.

• Double-click the tMysqlConnection component to display its Basic settings view.

• Set the database connection details manually or select Repository from the Property Type list and select your DB connection if it has already been defined and stored in the Metadata area of the Repository tree view.
For more information about Metadata, see *Talend Studio User Guide*.

- Double-click the first **tMysqlInput** component to display its **Basic settings** view.

- Select the **Use an existing connection** check box. If you are using more than one DB connection component in your Job, select the component you want to use from the **Component List**.

- Click the three-dot button next to **Edit schema** and define the data structure of the target table, or select **Repository** from the **Schema** list and select the target table if the schema has already been defined and stored in the **Metadata** area of the **Repository** tree view.

  In this scenario, we use built-in schemas.

- Define the columns as shown above, and then click **OK** to propagate the schema structure to the output component and close the schema dialog box.

- Fill the **Table Name** field with the name of the target table, *customer_info_merge* in this scenario.

- Click the **Guess Query** button, or type **"SELECT * FROM customer_info_merge"** in the **Query** area, to retrieve all the table columns.
• Define the properties of the second `tMysqlInput` component, using exactly the same settings as for the first `tMysqlInput` component.

• In the Basic settings view of each `tLogRow` component, select the Table option in the Mode area so that the contents will be displayed in table cells on the console.

• Double-click the `tSQLTemplateMerge` component to display its Basic settings view.

• Type in the names of the source table and the target table in the relevant fields.

In this scenario, the source table is `new_customer_info`, which contains eight records; the target table is `customer_info_merge`, which contains five records, and both tables have the same data structure.

Note:

The source table and the target table may have different schema structures. In this case, however, make sure that the source column and target column specified in each line of the Merge ON table, the UPDATE Columns table, and the INSERT Columns table are identical in data type and the target column length allows the insertion of the data from the corresponding source column.

• Define the source schema manually, or select Repository from the Schema list and select the relevant table if the schema has already been defined and stored in the Metadata area of the Repository tree view.

In this scenario, we use built-in schemas.
• Define the columns as shown above and click **OK** to close the schema dialog box, and do the same for the target schema.

• Click the green plus button beneath the **Merge ON** table to add a line, and select the **ID** column as the primary key.

```
<table>
<thead>
<tr>
<th>Column</th>
<th>Key</th>
<th>DB Type</th>
<th>N..</th>
<th>Length</th>
<th>Precision</th>
<th>Def...</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td></td>
<td>INT</td>
<td></td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>VARCHAR</td>
<td></td>
<td>13</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BirthDate</td>
<td></td>
<td>DATETIME</td>
<td></td>
<td>19</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td></td>
<td>VARCHAR</td>
<td></td>
<td>28</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>City</td>
<td></td>
<td>VARCHAR</td>
<td></td>
<td>14</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revenue</td>
<td></td>
<td>INT</td>
<td></td>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Warning:**

The columns defined as the primary key CANNOT and MUST NOT be made subject to updates.

• Select the **Specify UPDATE WHERE clause** check box and type in `customer_info_merge.ID >= 4` within double quotation marks in the **WHERE clause** field so that only those existing records with an ID equal to or greater than 4 will be updated.

```
Use UPDATE (WHEN MATCHED)
```

In this scenario, we want to update all the columns according to the customer IDs. Therefore, we select all the columns except the **ID** column.

**Warning:**

The columns defined as the primary key CANNOT and MUST NOT be made subject to updates.
• Select the **Use INSERT (WHEN MATCHED)** check box and define the columns to take data from and insert data to in the **INSERT Columns** table.

In this example, we want to insert all the records that do not exist in the target table.

• Select the **SQL Template** view to display and add the SQL templates to be used.

By default, the **SQLTemplateMerge** component uses two system SQL templates: **MergeUpdate** and **MergeInsert**.

**Note:**
In the **SQL Template** tab, you can add system SQL templates or create your own and use them within your Job to carry out the coded operation. For more information, see **tSQLTemplateFilterColumns Standard properties** on page 3669.
- Click the **Add** button to add a line and select **Commit** from the template list to commit the merge result to your database.

Alternatively, you can connect the **tSQLTemplateMerge** component to a **tSQLTemplateCommit** or **tMysqlCommit** component using a **Trigger > OnSubjobOK** connection to commit the merge result to your database.

- Save your Job and press **F6** to run it.

Both the original contents of the target table and the merge result are displayed on the console. In the target table, records No. 4 and No. 5 contain the updated information, and records No. 6 through No. 8 contain the inserted information.

```
Starting job MergeData at 10:56 01/04/2011

[statistics] connecting to socket on port 3580
[statistics] connected

+------------------+
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>BirthDate</th>
<th>Address</th>
<th>City</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Calvin Wilson</td>
<td>06-04-2008</td>
<td>997 East Calle Primera</td>
<td>Topeka</td>
<td>935</td>
</tr>
<tr>
<td>2</td>
<td>Benjamin Johnson</td>
<td>08-01-2007</td>
<td>306 Newbury Road</td>
<td>Olympia</td>
<td>042</td>
</tr>
<tr>
<td>3</td>
<td>William Hayes</td>
<td>03-05-2007</td>
<td>415 San Marcos</td>
<td>Carson City</td>
<td>365</td>
</tr>
<tr>
<td>4</td>
<td>George Quincy</td>
<td>01-05-2000</td>
<td>516 Carpinteria South</td>
<td>Bismarck</td>
<td>1997</td>
</tr>
<tr>
<td>5</td>
<td>Ronald Coleidge</td>
<td>12-01-2009</td>
<td>789 Newbury Road</td>
<td>Jackson</td>
<td>6516</td>
</tr>
</tbody>
</table>

[statistics] disconnected
```

**Job MergeData ended at 10:56 01/04/2011. [exit code=0]**
tSQLTemplateRollback

Cancels the transaction committed in the SQLTemplate database.

**tSQLTemplateRollback Standard properties**

These properties are used to configure tSQLTemplateRollback running in the Standard Job framework.

The Standard tSQLTemplateRollback component belongs to the ELT family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th><strong>Database Type</strong></th>
<th>Select the database type you want to connect to from the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Component List</strong></td>
<td>Select the ELT database connection component in the list if more than one connection is planned for the current Job.</td>
</tr>
<tr>
<td><strong>Close Connection</strong></td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

### Advanced settings

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

### Global Variables

| **Global Variables** | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is more commonly used with the \texttt{tSQLTemplateCommit} component and the ELT database connection component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Template</td>
<td>SQL Template List</td>
</tr>
</tbody>
</table>

Related scenarios

For a \texttt{tSQLTemplateRollback} related scenario, see \texttt{Scenario: Filtering and aggregating table columns directly on the DBMS} on page 3663.
tSqoopExport

Defines the arguments required by Sqoop for transferring data to a RDBMS.

tSqoopExport calls sqoop to transfer data from the Hadoop Distributed File System (HDFS) to a relational database management system (RDBMS).

Please be aware that some features provided by this component are only supported by the latest Sqoop version. For further information about the availability of each feature, see Apache’s documentation about Sqoop.

Note:
Sqoop is typically installed in every Hadoop distribution. But if the Hadoop distribution you need to use have no Sqoop installed, you have to install one on your own and ensure to add the Sqoop command line to the PATH variable of that distribution. For further information about how to install Sqoop, see the documentation of Sqoop.

Additional arguments

<table>
<thead>
<tr>
<th>Commandline mode</th>
<th>Java API mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>--driver</td>
<td>jdbc.driver.class</td>
</tr>
<tr>
<td>--direct-split-size</td>
<td>import.direct.split.size</td>
</tr>
<tr>
<td>--inline-lob-limit</td>
<td>import.max.inline.lob.size</td>
</tr>
<tr>
<td>--split-by</td>
<td>db.split.column</td>
</tr>
<tr>
<td>--warehouse-dir</td>
<td>hdfs.warehouse.dir</td>
</tr>
<tr>
<td>--enclosed-by</td>
<td>codegen.output.delimiters.enclose</td>
</tr>
<tr>
<td>--escaped-by</td>
<td>codegen.output.delimiters.escape</td>
</tr>
<tr>
<td>--fields-terminated-by</td>
<td>codegen.output.delimiters.field</td>
</tr>
<tr>
<td>--lines-terminated-by</td>
<td>codegen.output.delimiters.record</td>
</tr>
<tr>
<td>--optionally-enclosed-by</td>
<td>codegen.output.delimiters.required</td>
</tr>
<tr>
<td>--input-enclosed-by</td>
<td>codegen.input.delimiters.enclose</td>
</tr>
<tr>
<td>--input-escaped-by</td>
<td>codegen.input.delimiters.escape</td>
</tr>
<tr>
<td>--input-fields-terminated-by</td>
<td>codegen.input.delimiters.field</td>
</tr>
<tr>
<td>--input-lines-terminated-by</td>
<td>codegen.input.delimiters.record</td>
</tr>
<tr>
<td>--input-optionally-enclosed-by</td>
<td>codegen.input.delimiters.required</td>
</tr>
<tr>
<td>--hive-home</td>
<td>hive.home</td>
</tr>
</tbody>
</table>
**tSqoopExport Standard properties**

These properties are used to configure tSqoopExport running in the Standard Job framework.

The Standard tSqoopExport component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Select the mode in which Sqoop is called in a Job execution.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use Commandline</strong></td>
<td>the Sqoop shell is used to call Sqoop. In this mode, you have to deploy and run the Job in the host where Sqoop is installed. Therefore, if you are a subscription-based user, we recommend installing and using a Jobserver provided by Talend in that host to run the Job; if you are using one of the Talend solutions with Big Data, you have to ensure that the Studio and the Sqoop to be used are in the same machine.</td>
</tr>
<tr>
<td><strong>Use Java API</strong></td>
<td>the Java API is used to call Sqoop. In this mode, the Job can be run locally in the Studio but you need to configure the connection to the Hadoop distribution to be used. Note that JDK is required to execute the Job in the Java API mode and the versions of the JDK kits installed in both machines must be compatible with each other; for example, the versions are the same or the JDK version of the Hadoop machine is more recent.</td>
</tr>
</tbody>
</table>

---

For further information about the arguments available in the Sqoop commandline mode, see the documentation of Sqoop.

The arguments listed earlier for the Java API mode are subject to updates and changes by Java. For further information about these arguments, see [http://svn.apache.org/repos/asf/sqoop/trunk/src/java/org/apache/sqoop/SqoopOptions.java](http://svn.apache.org/repos/asf/sqoop/trunk/src/java/org/apache/sqoop/SqoopOptions.java)
### Hadoop properties

Either **Built-in** or **Repository**:

- **Built-in**: you enter the configuration information of the Hadoop distribution to be used locally for this component only.

- **Repository**: you have already created the Hadoop connection and stored it in the Repository; therefore, you reuse it directly for the component configuration and the Job design. For further information about how to create a centralized Hadoop connection, see [Talend Open Studio for Big Data Getting Started Guide](https://help.talend.com).

### Distribution

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this Distribution drop-down list, the **Microsoft HD Insight** option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select **Amazon EMR**, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The **Custom** option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.
Note that custom versions are not officially supported by **Talend**. **Talend** and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.

**Note:**
In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see [Connecting to a custom Hadoop distribution](#) on page 1589.

<table>
<thead>
<tr>
<th>Hadoop Version</th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NameNode URI</td>
<td>Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called <em>masternode</em> as the NameNode, then the location is hdfs://masternode:portnumber. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL, the scheme should be swebhdfs and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS.</td>
</tr>
</tbody>
</table>
| JobTracker Host | Select this check box and in the displayed field, enter the location of the ResourceManager of your distribution. For example, tal-qa114.tale nd.lan:8050. This property is required when the query you want to use is executed in Windows and it is a Select query. For example, `SELECT your_column_name FROM your_table_name` Then you can continue to set the following parameters depending on the configuration of the Hadoop cluster to be used (if you leave the check box of a parameter clear, then at runtime, the configuration about this parameter in the Hadoop cluster to be used will be ignored):  
  1. Select the **Set resourcemanager scheduler address** check box and enter the Scheduler address in the field that appears.  
  2. Select the **Set jobhistory address** check box and enter the location of the JobHistory server of the |
Hadoop cluster to be used. This allows the metrics information of the current Job to be stored in that JobHistory server.

3. Select the Set staging directory check box and enter this directory defined in your Hadoop cluster for temporary files created by running programs. Typically, this directory can be found under the yarn.app.mapreduce.am.staging-dir property in the configuration files such as yarn-site.xml or mapred-site.xml of your distribution.

4. Allocate proper memory volumes to the Map and the Reduce computations and the ApplicationMaster of YARN by selecting the Set memory check box in the Advanced settings view.

5. Select the Set Hadoop user check box and enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed.

6. Select the Use datanode hostname check box to allow the Job to access datanodes via their hostnames. This actually sets the dfs.client.use.datanode.hostname property to true. When connecting to a S3N filesystem, you must select this check box.

For further information about these parameters, see the documentation or contact the administrator of the Hadoop cluster to be used.

For further information about the Hadoop Map/Reduce framework, see the Map/Reduce tutorial in Apache’s Hadoop documentation on http://hadoop.apache.org.

Use kerberos authentication

If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the Force MapR ticket authentication check box and the Use Kerberos authentication check box clear, and then MapR should be able to automatically find that ticket on the fly.

In addition, since this component performs Map/Reduce computations, you also need to authenticate the related services such as the Job history server and the Resource

Use kerberos authentication
manager or Jobtracker depending on your distribution in the corresponding field. These principals can be found in the configuration files of your distribution. For example, in a CDH4 distribution, the Resource manager principal is set in the `yarn-site.xml` file and the Job history principal in the `mapred-site.xml` file.

This check box is available depending on the Hadoop distribution you are connecting to.

| Use a keytab to authenticate | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is **user1** and the principal to be used is **guest**; in this situation, ensure that **user1** has the right to read the keytab file to be used. |
| Hadoop user name | Enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed. Note that this field is available depending on the distribution you are using. |
| JDBC property | Either **Built-in** or **Repository**:

- **Built-in**: you enter the connection information of the database to be used locally for this component only.

- **Repository**: you have already created the database connection and stored it in the **Repository**; therefore, you reuse it directly for the component configuration and the Job design. For further information about how to create a centralized database connection, see **Talend Studio User Guide**.

Note that only the **General JDBC** connection stored in the **Repository** is supported. |
| Connection | Type in the JDBC URL used to connect to the target database. |
| User name and Password | Type in the authentication information to the target database.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.
If your password is stored in a file, select the **The password is stored in a file** check box and enter the path to that file in the **File path** field that is displayed.

- This file can be stored either in the machine where the Job is to be executed or in the HDFS system of the Hadoop cluster to be used.
- The password stored in this file must not contain `\n` (the newline escape) at the end, that is to say, you must not insert a new line at the end of the password even though this line is empty.

Note that this feature is available depending on the Sqoop version you are using.

<table>
<thead>
<tr>
<th><strong>Driver JAR</strong></th>
<th>In either the <strong>Use Commandline</strong> mode or the <strong>Java API</strong> mode, you must add the driver file of the database to be used to the <strong>lib</strong> folder of the Hadoop distribution you are using. For that purpose, use this <strong>Driver JAR</strong> table to add that driver file for the current Job you are designing.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table Name</strong></td>
<td>Type in the name of the target table to which data is transferred from HDFS. This table must already exist in the target database. The input files are read and parsed into a set of records according to the user-specified delimiters.</td>
</tr>
<tr>
<td><strong>Export Dir</strong></td>
<td>Enter the path to the source data to be transferred in HDFS.</td>
</tr>
<tr>
<td><strong>Direct</strong></td>
<td>Select this check box to use the export fast path.</td>
</tr>
<tr>
<td><strong>Specify Number of Mappers</strong></td>
<td>Select this check box to indicate the number of map tasks (parallel processes) used to perform the data transfer. If you do not want Sqoop to work in parallel, enter 1 in the displayed field.</td>
</tr>
<tr>
<td><strong>Call a stored procedure</strong></td>
<td>Select this check box to enable the component to call a specific store procedure to write data into the target database. You need to enter the name of the stored procedure to be used in the field that is displayed. Syntax and capabilities of stored procedures vary among different databases; for this reason, you are recommended to consult the documentation of the database you want to use for details about how a given stored procedure could be called.</td>
</tr>
<tr>
<td><strong>Use batch mode</strong></td>
<td>Select this check box to execute the statements in batch mode instead of running a multi-row INSERT statement to write multiple records in the target database.</td>
</tr>
<tr>
<td><strong>Clear staging table</strong></td>
<td>If you are using a specific staging table for the desired data transfer, select this check box to ensure that the staging table is empty when the data transfer runs.</td>
</tr>
</tbody>
</table>
### Define a staging table

Select this check box to create a staging table for the data to be transferred. The transferred data is staged within this table before being written into the target table so as to avoid only a part of the data being committed to the target table when the transfer fails.

For further information about whether a staging table is supported for a given data transfer, see Apache’s documentation for Sqoop.

### Specify how updates are performed when new rows are found with non-match keys in database

Select this check box to determine the action to be taken when a given update key does not have any matching records in the target table. You can then select either of the following options:

- **Update only**: this updates only the records that already exist in the target table.
- **Allow insert**: this works like the SQL UPSERT statement. It writes new records in the table if they do not exist there.

### Use column for update

Select this check box and in the table that is displayed, add the columns to be used as the update key.

### Print Log

Select this check box to activate the **Verbose** check box.

### Verbose

Select this check box to print more information while working, for example, the debugging information.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use MySQL default delimiters</strong></td>
<td>Select this check box to use MySQL’s default delimiter set. This check box is available only to the Commandline mode.</td>
</tr>
<tr>
<td><strong>Define Java mapping</strong></td>
<td>Sqoop provides default configuration that maps most SQL types to appropriate Java types. If you need to use your custom map to overwrite the default ones at runtime, select this check box and define the map(s) you want to use in the table that appears.</td>
</tr>
<tr>
<td><strong>Additional arguments</strong></td>
<td>Complete this table to use additional arguments if needed. By adding additional arguments, you are able to perform multiple operations in one single transaction. For example, you can use <code>--hive-import</code> and <code>--hive-table</code> in the Commandline mode or <code>hive.import</code> and <code>hive.table.name</code> in the Java API mode to create Hive table and write data in at the runtime of the transaction writing data in HDFS. For further information about the available Sqoop arguments in the Commandline mode and the Java API mode, respectively, see Additional arguments on page 3684.</td>
</tr>
<tr>
<td><strong>Use speed parallel data transfers</strong></td>
<td>Select this check box to enable quick parallel data transfers between the Teradata database and the Hortonworks Hadoop distribution. Then the Specific pa</td>
</tr>
</tbody>
</table>
rams table and the **Use additional params** check box appear to allow you to specify the Teradata parameters required by parallel transfers.

- In the **Specific params** table, two columns are available:
  - **Argument**: select the parameters as needed from the drop-down list. They are the most common parameters for the parallel transfer.
  - **Value**: type in the value of the parameters.

- By selecting the **Additional params** check box, you make the **Specific additional params** field displayed. In this field, you can enter the Teradata parameters that you need to use but are not provided in the **Specific params** table. The syntax for a parameter is `-Dparameter=value` and when you put more than one parameter in this field, separate them using whitespace.

You must ensure that the Hortonworks Connector for Teradata has been installed in your Hortonworks cluster. The latest connector can be downloaded from the website of Hortonworks and installed by following the explanations from [http://hortonworks.com/wp-content/uploads/2014/02/bk_HortonworksConnectorForTeradata.pdf](http://hortonworks.com/wp-content/uploads/2014/02/bk_HortonworksConnectorForTeradata.pdf). In the same document, you can as well find the detailed explanations about each parameter that is available for the parallel transfer purpose.

Available in the **Use Commandline** mode only.

### Hadoop properties

**Talend Studio** uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache's Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such as [http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml](http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml).
Apache also provides a page to list the Hive-related properties: https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties.

Mapred job map memory mb and Mapred job reduce memory mb
You can tune the map and reduce computations by selecting the Set memory check box to set proper memory allocations for the computations to be performed by the Hadoop system.

In that situation, you need to enter the values you need in the Mapred job map memory mb and the Mapred job reduce memory mb fields, respectively. By default, the values are both 1000 which are normally appropriate for running the computations.

The memory parameters to be set are Map (in Mb), Reduce (in Mb) and ApplicationMaster (in Mb). These fields allow you to dynamically allocate memory to the map and the reduce computations and the ApplicationMaster of YARN.

Path separator in server
Leave the default value of the Path separator in server as it is, unless you have changed the separator used by your Hadoop distribution’s host machine for its PATH variable or in other words, that separator is not a colon (:). In that situation, you must change this value to the one you are using in that host.

tStatCatcher Statistics
Select this check box to collect log data at the component level.

Global Variables

Global Variables

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

EXIT_CODE: the exit code of the remote command. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule
This component is used standalone. It respects the Sqoop prerequisites. You need necessary knowledge about Sqoop to use it.

We recommend using the Sqoop of version 1.4+ in order to benefit the full functions of these components.
Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

Limitation

If you have selected the Use Commandline mode, you need to use the host where Sqoop is installed to run the Job using this component.

Connections

Outgoing links (from this component to another):

**Trigger**: Run if; On Subjob Ok; On Subjob Error.

Incoming links (from one component to this one):

**Row**: Iterate;

**Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error

For further information regarding connections, see Talend Studio User Guide.

Related scenarios

No scenario is available for the Standard version of this component yet.
tSqoopImport

Defines the arguments required by Sqoop for writing the data of your interest into HDFS.

tSqoopImport calls Sqoop to transfer data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS).

Note:
Sqoop is typically installed in every Hadoop distribution. But if the Hadoop distribution you need to use have no Sqoop installed, you have to install one on your own and ensure to add the Sqoop command line to the PATH variable of that distribution. For further information about how to install Sqoop, see the documentation of Sqoop.

tSqoopImport Standard properties

These properties are used to configure tSqoopImport running in the Standard Job framework.
The Standard tSqoopImport component belongs to the Big Data and the File families.
The component in this framework is available when you are using one of the Talend solutions with Big Data.

Basic settings

<table>
<thead>
<tr>
<th>Mode</th>
<th>Select the mode in which Sqoop is called in a Job execution.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Commandline</td>
<td>the Sqoop shell is used to call Sqoop. In this mode, you have to deploy and run the Job in the host where Sqoop is installed. Therefore, if you are a subscription-based user, we recommend installing and using a Jobserver provided by Talend in that host to run the Job; if you are using one of the Talend solutions with Big Data, you have to ensure that the Studio and the Sqoop to be used are in the same machine.</td>
</tr>
<tr>
<td>Use Java API</td>
<td>the Java API is used to call Sqoop. In this mode, the Job can be run locally in the Studio but you need to configure the connection to the Hadoop distribution to be used. Note that JDK is required to execute the Job in the Java API mode and the versions of the JDK kits installed in both machines must be compatible with each other; for example, the versions are the same or the JDK version of the Hadoop machine is more recent.</td>
</tr>
</tbody>
</table>

Hadoop properties

Either Built-in or Repository:
- Built-in: you enter the configuration information of the Hadoop distribution to be used locally for this component only.
- Repository: you have already created the Hadoop connection and stored it in the Repository; therefore, you reuse it directly for the component configuration and the Job design. For further information about how to create a centralized
### Distribution

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones require specific configuration:

- If available in this Distribution drop-down list, the **Microsoft HD Insight** option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center ([https://help.talend.com](https://help.talend.com)).

- If you select **Amazon EMR**, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

- The **Custom** option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

   In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this [Hadoop configuration list](https://help.talend.com) and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.
### Note:
In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see Connecting to a custom Hadoop distribution on page 1589.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hadoop Version</strong></td>
<td>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</td>
</tr>
<tr>
<td><strong>NameNode URI</strong></td>
<td>Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called <code>masternode</code> as the NameNode, then the location is <code>hdfs://masternode:portnumber</code>. If you are using WebHDFS, the location should be <code>webhdfs://masternode:portnumber</code>; if this WebHDFS is secured with SSL, the scheme should be <code>swebhdfs</code> and you need to use a <code>LibraryLoad</code> in the Job to load the library required by the secured WebHDFS.</td>
</tr>
</tbody>
</table>
| **JobTracker Host**    | Select this check box and in the displayed field, enter the location of the ResourceManager of your distribution. For example, `tal-qall4.taled.lan:8050`. This property is required when the query you want to use is executed in Windows and it is a Select query. For example, `SELECT your_column_name FROM your_table_name` Then you can continue to set the following parameters depending on the configuration of the Hadoop cluster to be used (if you leave the check box of a parameter clear, then at runtime, the configuration about this parameter in the Hadoop cluster to be used will be ignored):

1. Select the **Set resourcemanager scheduler address** check box and enter the Scheduler address in the field that appears.
2. Select the **Set jobhistory address** check box and enter the location of the JobHistory server of the Hadoop cluster to be used. This allows the metrics information of the current Job to be stored in that JobHistory server.
3. Select the **Set staging directory** check box and enter this directory defined in your Hadoop cluster for temporary files created by running programs. Typically, this directory can be found under the `yarn.app.mapreduce.am.staging-dir` property in the configuration files such as `yarn-site.xml` or `mapred-site.xml` of your distribution.
4. Allocate proper memory volumes to the **Map** and the **Reduce** computations and the **ApplicationMaster** of YARN by selecting the **Set memory** check box in the **Advanced settings** view.

5. Select the **Set Hadoop user** check box and enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed.

6. Select the **Use datanode hostname** check box to allow the Job to access datanodes via their hostnames. This actually sets the `dfs.client.use.datanode.hostname` property to `true`. When connecting to a S3N filesystem, you must select this check box.

For further information about these parameters, see the documentation or contact the administrator of the Hadoop cluster to be used.


### Use kerberos authentication

If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the **Force MapR ticket authentication** check box and the **Use Kerberos authentication** check box clear, and then MapR should be able to automatically find that ticket on the fly.

In addition, since this component performs Map/Reduce computations, you also need to authenticate the related services such as the Job history server and the Resource manager or Jobtracker depending on your distribution in the corresponding field. These principals can be found in the configuration files of your distribution. For example, in a CDH4 distribution, the Resource manager principal is set in the `yarn-site.xml` file and the Job history principal in the `mapred-site.xml` file.

This check box is available depending on the Hadoop distribution you are connecting to.
| **Use a keytab to authenticate** | Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is *user1* and the principal to be used is *guest*; in this situation, ensure that *user1* has the right to read the keytab file to be used. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hadoop user name</strong></td>
<td>Enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed. Note that this field is available depending on the distribution you are using.</td>
</tr>
</tbody>
</table>
| **JDBC property** | Either **Built-in** or **Repository**:

- **Built-in**: you enter the connection information of the database to be used locally for this component only.

- **Repository**: you have already created the database connection and stored it in the **Repository**; therefore, you reuse it directly for the component configuration and the Job design. For further information about how to create a centralized database connection, see *Talend Studio User Guide*.

Note that only the **General JDBC** connection stored in the **Repository** is supported. |
| **Connection** | Enter the JDBC URL used to connect to the database where the source data is stored. |
| **User name and Password** | Enter the authentication information used to connect to the source database.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

If your password is stored in a file, select the **The password is stored in a file** check box and enter the path to that file in the **File path** field that is displayed.

- This file can be stored either in the machine where the Job is to be executed or in the HDFS system of the Hadoop cluster to be used.

- The password stored in this file must not contain \n (the newline escape) at the end, that is to say, you must not insert a new line at the end of the password even though this line is empty. |
Note that this feature is available depending on the Sqoop version you are using.

| **Driver JAR** | In either the **Use Commandline** mode or the **Java API** mode, you must add the driver file of the database to be used to the \lib\ folder of the Hadoop distribution you are using. For that purpose, use this **Driver JAR** table to add that driver file for the current Job you are designing. |
| **Table Name** | Type in the name of the table to be transferred into HDFS. This field is not available when you are using the free-form query mode by selecting the **Use query** check box. |
| **File format** | Select a file format for the data to be transferred:  
- **textfile**  
- **sequencefile**  
- **Avro file**  
- **Parquet file**: the version of Sqoop must be 1.4.6. |
| **Delete target directory** | Select this check box to remove the target directory of the transfer. |
| **Append** | Select this check box to append transferred data to an existing dataset in HDFS. |
| **Compress** | Select this check box to enable compression. |
| **Direct** | Select this check box to use the import fast path. |
| **Specify columns** | Select this check box to display the column table where you can specify the columns you want to transfer into HDFS. |
| **Use WHERE clause** | Select this check box to use a WHERE clause that controls the rows to be transferred. In the field displayed, you can type in the condition used to select the rows you want. For example, type `id >400` to import only the rows where the `id` column has a value greater than 400. |
| **Use query** | Select this check box to use the free-form query mode provided by Sqoop.  
Once selecting it, you are able to enter the free-form query you need to use.  
Then, you must specify the target directory and if the Sqoop imports data in parallel, specify as well the **Split by** argument.  

**Warning:**  
Once queries are entered here, the value of the argument `--fields-terminated-by` can only be set to `\'\'` in the **Additional arguments** table in the **Advanced settings** tab. |
Specify Target Dir
Select this check box to enter the path to the target location, in HDFS, where you want to transfer the source data to.
This location should be a new directory; otherwise, you must select the Append check box.

Specify Split by
Select this check box, then, enter the table column you need and are able to use as the splitting column to split the workload.
For example, for a table where the id column is the key column, enter tablename.id. Then Sqoop will split the data to be transferred according to their ID values and imports them in parallel.

Specify Number of Mappers
Select this check box to indicate the number of map tasks (parallel processes) used to perform the data transfer.
If you do not want Sqoop to work in parallel, enter 1 in the displayed field.

Print Log
Select this check box to activate the Verbose check box.

Verbose
Select this check box to print more information while working, for example, the debugging information.

Advanced settings

Use MySQL default delimiters
Select this check box to use MySQL’s default delimiter set. This check box is available only to the Commandline mode.

Define Java mapping
Sqoop provides default configuration that maps most SQL types to appropriate Java types. If you need to use your custom map to overwrite the default ones at runtime, select this check box and define the map(s) you want to use in the table that appears.

Define Hive mapping
Sqoop provides default configuration that maps most SQL types to appropriate Hive types. If you need to use your custom map to overwrite the default ones at runtime, select this check box and define the map(s) you want to use in the table that appears.

Additional arguments
Complete this table to use additional arguments if needed be.
By adding additional arguments, you are able to perform multiple operations in one single transaction. For example, you can use --hive-import and --hive-table in the Commandline mode or hive.import and hive.table.name in the Java API mode to create Hive table and write data in at the runtime of the transaction writing data in HDFS. For further information about the availableSqoop arguments in the Commandline mode and the Java API mode, respectively, see Additional arguments on page 3684.
| Use speed parallel data transfers | Select this check box to enable quick parallel data transfers between the Teradata database and the Hortonworks Hadoop distribution. Then the **Specific params** table and the **Use additional params** check box appear to allow you to specify the Teradata parameters required by parallel transfers.

- In the **Specific params** table, two columns are available:
  - **Argument**: select the parameters as needed from the drop-down list. They are the most common parameters for the parallel transfer.
  - **Value**: type in the value of the parameters.
- By selecting the **Additional params** check box, you make the **Specific additional params** field displayed. In this field, you can enter the Teradata parameters that you need to use but are not provided in the **Specific params** table. The syntax for a parameter is `-Dparameter=value` and when you put more than one parameter in this field, separate them using whitespace.

You must ensure that the Hortonworks Connector for Teradata has been installed in your Hortonworks cluster. The latest connector can be downloaded from the website of Hortonworks and installed by following the explanations from [http://hortonworks.com/wp-content/uploads/2014/02/bk_HortonworksConnectorForTeradata.pdf](http://hortonworks.com/wp-content/uploads/2014/02/bk_HortonworksConnectorForTeradata.pdf). In the same document, you can as well find the detailed explanations about each parameter that is available for the parallel transfer purpose.

Available in the **Use Commandline** mode only.

| Hadoop properties | **Talend Studio** uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.

- Note that if you are using the centrally stored metadata from the **Repository**, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the **Property type** from **Repository** to **Built-in**.

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on [http://hadoop.apache.org/docs](http://hadoop.apache.org/docs) and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the `hdfs-default.xml` file of your distribution, such
A Mapred job map memory mb and Mapred job reduce memory mb can be tuned by selecting the Set memory check box to set proper memory allocations for the computations to be performed by the Hadoop system. In that situation, you need to enter the values you need in the Mapred job map memory mb and the Mapred job reduce memory mb fields, respectively. By default, the values are both 1000 which are normally appropriate for running the computations.

The memory parameters to be set are Map (in Mb), Reduce (in Mb) and ApplicationMaster (in Mb). These fields allow you to dynamically allocate memory to the map and the reduce computations and the ApplicationMaster of YARN.

Path separator in server

Leave the default value of the Path separator in server as it is, unless you have changed the separator used by your Hadoop distribution’s host machine for its PATH variable or in other words, that separator is not a colon (:). In that situation, you must change this value to the one you are using in that host.

Global Variables

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

EXIT_CODE: the exit code of the remote command. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

This component is used standalone. It respects the Sqoop prerequisites. You need necessary knowledge about Sqoop to use it.
We recommend using the Sqoop of version 1.4+ in order to benefit the full functions of these components.

For further information about Sqoop, see the Sqoop manual on: [http://sqoop.apache.org/docs/](http://sqoop.apache.org/docs/)

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with *Talend Studio*. The following list presents MapR related information for example:

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR's documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

- Set the `-Djava.library.path` argument, for example, in the *Job Run VM arguments* area of the *Run/Debug* view in the *Preferences* dialog box in the *Window* menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the *Data viewer* to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

### Limitation

If you have selected the *Use Commandline* mode, you need to use the host where Sqoop is installed to run the Job using this component.

### Connections

Outgoing links (from this component to another):

**Trigger**: Run if; On Subjob Ok; On Subjob Error.

Incoming links (from one component to this one):

**Row**: Iterate;

**Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error

For further information regarding connections, see *Talend Studio  User Guide*. 
Scenario: Importing a MySQL table to HDFS

This scenario applies only to a Talend solution with Big Data.

This scenario illustrates how to use `tSqoopImport` to import a MySQL table to a given HDFS system.

The sample data to be used in this scenario reads as follows:

<table>
<thead>
<tr>
<th>id, wage, mod_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 2000, 2008-06-26 04:25:59</td>
</tr>
<tr>
<td>1, 2300, 2011-06-12 05:29:45</td>
</tr>
<tr>
<td>3, 3000, 2010-05-02 15:34:05</td>
</tr>
</tbody>
</table>

The data is stored in a MySQL table called `sqoopmerge`.

Before starting to replicate this scenario, ensure that you have appropriate rights and permissions to access the Hadoop distribution to be used. Then proceed as follows:

### Dropping the component

**Procedure**

1. In the **Integration** perspective of the Studio, create an empty Job from the **Job Designs** node in the **Repository** tree view.
   
   For further information about how to create a Job, see the *Talend Studio User Guide*.

2. Drop `tSqoopImport` onto the workspace.

### Importing the MySQL table

**About this task**

Configuring `tSqoopImport`  

**Procedure**

1. Double-click `tSqoopImport` to open its **Component** view.
2. In the **Mode** area, select **Use Java API**.

3. In the **Version** area, select the Hadoop distribution to be used and its version. If you cannot find from the list the distribution corresponding to yours, select **Custom** so as to connect to a Hadoop distribution not officially supported in the Studio.

   For a step-by-step example about how to use this **Custom** option, see [Connecting to a custom Hadoop distribution](#) on page 1589.

4. In the **NameNode URI** field, enter the location of the master node, the NameNode, of the distribution to be used. If you are using WebHDFS, the location should be `webhdfs://master:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS.
5. In the **Resource Manager** field, enter the location of the ResourceManager of your distribution. Note that the notion Job in this term JobTracker designates the MR or the MapReduce jobs described in Apache's documentation on [http://hadoop.apache.org/](http://hadoop.apache.org/).

6. If the distribution to be used requires Kerberos authentication, select the **Use Kerberos authentication** check box and complete the authentication details. Otherwise, leave this check box clear.

   If you need to use a Kerberos keytab file to log in, select **Use a keytab to authenticate**. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

   Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is *user1* and the principal to be used is *guest*; in this situation, ensure that *user1* has the right to read the keytab file to be used.

7. In the **Connection** field, enter the URI of the MySQL database where the source table is stored. For example, `jdbc:mysql://10.42.10.13/mysql`.

8. In **Username** and **Password**, enter the authentication information.

9. Under the **Driver JAR** table, click the `[+]` button to add one row, then in this row, click the `...` button to display the drop-down list and select the jar file to be used from that list. In this scenario, it is `mysql-connector-java-5.1.30-bin.jar`.

   If the `...` button does not appear, click anywhere in this row to make it displayed.

10. In the **Table Name** field, enter the name of the source table. In this scenario, it is `sqoopmerge`.

11. From the **File format** list, select the format that corresponds to the data to be used, **textfile** in this scenario.

12. Select the **Specify target dir** check box and enter the directory where you need to import the data to. For example, `/user/ychen/target_old`.

### Configuring tSqoopImport

#### Procedure

1. Double-click **tSqoopImport** to open its **Component** view.
2. In the Mode area, select Use Java API.

3. In the Version area, select the Hadoop distribution to be used and its version. If you cannot find from the list the distribution corresponding to yours, select Custom so as to connect to a Hadoop distribution not officially supported in the Studio.

   For a step-by-step example about how to use this Custom option, see Connecting to a custom Hadoop distribution on page 1589.

4. In the NameNode URI field, enter the location of the master node, the NameNode, of the distribution to be used. For example, hdfs://talend-cdh4-namenode:8020. If you are using WebHDFS, the location should be webhdfs://masternode:portnumber; if this WebHDFS is secured with SSL,
the scheme should be swebhdfs and you need to use a tLibraryLoad in the Job to load the library required by the secured WebHDFS.

5. In the JobTracker Host field, enter the location of the JobTracker of your distribution. For example, talend-cdh4-namenode:8021.

Note that the notion Job in this term JobTracker designates the MR or the MapReduce jobs described in Apache's documentation on http://hadoop.apache.org/.

6. If the distribution to be used requires Kerberos authentication, select the Use Kerberos authentication check box and complete the authentication details. Otherwise, leave this check box clear.

If you need to use a Kerberos keytab file to log in, select Use a keytab to authenticate. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the Principal field and the access path to the keytab file itself in the Keytab field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is user1 and the principal to be used is guest; in this situation, ensure that user1 has the right to read the keytab file to be used.

7. In the Connection field, enter the URI of the MySQL database where the source table is stored. For example, jdbc:mysql://10.42.10.13/mysql.

8. In Username and Password, enter the authentication information.

9. Under the Driver JAR table, click the [+] button to add one row, then in this row, click the [...] button to display the drop-down list and select the jar file to be used from that list. In this scenario, it is mysql-connector-java-5.1.30-bin.jar.

If the [...] button does not appear, click anywhere in this row to make it displayed.

10. In the Table Name field, enter the name of the source table. In this scenario, it is sqoopmerge.

11. From the File format list, select the format that corresponds to the data to be used, textfile in this scenario.

12. Select the Specify target dir check box and enter the directory where you need to import the data to. For example, /user/ychen/target_old.

**Executing the Job**

Then you can press F6 to run this Job.

Once done, you can verify the results in the target directory you have specified, in the web console of the Hadoop distribution used.
Contents of directory /user/ychen/target_old

Goto: [user/ychen/target_old]  go

Go to parent directory

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Size</th>
<th>Replication</th>
<th>Block Size</th>
<th>Modification Time</th>
<th>Permission</th>
<th>Owner</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>_SUCCESS</td>
<td>file</td>
<td>0 KB</td>
<td>3</td>
<td>64 MB</td>
<td>2013-10-14 20:28</td>
<td>rw-r--r--</td>
<td>ychen</td>
<td>ychen</td>
</tr>
<tr>
<td>_logs</td>
<td>dir</td>
<td></td>
<td></td>
<td></td>
<td>2013-10-14 20:27</td>
<td>rwxr-xr-x</td>
<td>ychen</td>
<td>ychen</td>
</tr>
<tr>
<td>part-m-00000</td>
<td>file</td>
<td>0.03 KB</td>
<td>3</td>
<td>64 MB</td>
<td>2013-10-14 20:27</td>
<td>rw-r--r--</td>
<td>ychen</td>
<td>ychen</td>
</tr>
<tr>
<td>part-m-00001</td>
<td>file</td>
<td>0.03 KB</td>
<td>3</td>
<td>64 MB</td>
<td>2013-10-14 20:28</td>
<td>rw-r--r--</td>
<td>ychen</td>
<td>ychen</td>
</tr>
<tr>
<td>part-m-00002</td>
<td>file</td>
<td>0.03 KB</td>
<td>3</td>
<td>64 MB</td>
<td>2013-10-14 20:28</td>
<td>rw-r--r--</td>
<td>ychen</td>
<td>ychen</td>
</tr>
<tr>
<td>part-m-00003</td>
<td>file</td>
<td>0.03 KB</td>
<td>3</td>
<td>64 MB</td>
<td>2013-10-14 20:28</td>
<td>rw-r--r--</td>
<td>ychen</td>
<td>ychen</td>
</tr>
</tbody>
</table>

Go back to DFS home

If you need to obtain more details about the Job, it is recommended to use the web console of the Jobtracker provided by the Hadoop distribution you are using.
**tSqoopImportAllTables**

Defines the arguments required by Sqoop for writing all of the tables of a database into HDFS.

*tSqoopImportAllTables* calls Sqoop to transfer all of the tables of a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS).

**Note:**
Sqoop is typically installed in every Hadoop distribution. But if the Hadoop distribution you need to use have no Sqoop installed, you have to install one on your own and ensure to add the Sqoop command line to the PATH variable of that distribution. For further information about how to install Sqoop, see the documentation of Sqoop.

**tSqoopImportAllTables Standard properties**

These properties are used to configure *tSqoopImportAllTables* running in the Standard Job framework. The Standard *tSqoopImportAllTables* component belongs to the Big Data and the File families. The component in this framework is available when you are using one of the *Talend* solutions with Big Data.

**Basic settings**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Select the mode in which Sqoop is called in a Job execution.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Commandline</td>
<td>the Sqoop shell is used to call Sqoop. In this mode, you have to deploy and run the Job in the host where Sqoop is installed. Therefore, if you are a subscription-based user, we recommend installing and using a Jobserver provided by <em>Talend</em> in that host to run the Job; if you are using one of the <em>Talend</em> solutions with Big Data, you have to ensure that the Studio and the Sqoop to be used are in the same machine.</td>
</tr>
<tr>
<td>Use Java API</td>
<td>the Java API is used to call Sqoop. In this mode, the Job can be run locally in the Studio but you need to configure the connection to the Hadoop distribution to be used. Note that JDK is required to execute the Job in the Java API mode and the versions of the JDK kits installed in both machines must be compatible with each other; for example, the versions are the same or the JDK version of the Hadoop machine is more recent.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hadoop properties</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Built-in</td>
<td>you enter the configuration information of the Hadoop distribution to be used locally for this component only.</td>
</tr>
<tr>
<td>• Repository</td>
<td>you have already created the Hadoop connection and stored it in the <strong>Repository</strong>; therefore, you reuse it directly for the component configuration and the Job design. For further information about how to create a centralized</td>
</tr>
</tbody>
</table>
Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following one requires specific configuration:

- If available in this Distribution drop-down list, the Microsoft HD Insight option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center (https://help.talend.com).

- If you select Amazon EMR, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center (https://help.talend.com).

- The Custom option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select Import from existing version to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select Import from zip to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

In Talend Exchange, members of Talend community have shared some ready-for-use configuration zip files which you can download from this Hadoop configuration list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the Import from existing version option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient Hadoop experience to handle any issues on your own.
<table>
<thead>
<tr>
<th><strong>Hadoop Version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NameNode URI</strong></td>
<td>Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called <code>masternode</code> as the NameNode, then the location is <code>hdfs://masternode:portnumber</code>. If you are using WebHDFS, the location should be <code>webhdfs://masternode:portnumber</code>; if this WebHDFS is secured with SSL, the scheme should be <code>swebhdfs</code> and you need to use a <code>LibraryLoad</code> in the Job to load the library required by the secured WebHDFS.</td>
</tr>
</tbody>
</table>
| **JobTracker Host** | Select this check box and in the displayed field, enter the location of the ResourceManager of your distribution. For example, `tal-qall4.taland.lan:8050`. This property is required when the query you want to use is executed in Windows and it is a Select query. For example, `SELECT your_column_name FROM your_table_name` Then you can continue to set the following parameters depending on the configuration of the Hadoop cluster to be used (if you leave the check box of a parameter clear, then at runtime, the configuration about this parameter in the Hadoop cluster to be used will be ignored):  
1. Select the **Set resourcemanager scheduler address** check box and enter the Scheduler address in the field that appears.  
2. Select the **Set jobhistory address** check box and enter the location of the JobHistory server of the Hadoop cluster to be used. This allows the metrics information of the current Job to be stored in that JobHistory server.  
3. Select the **Set staging directory** check box and enter this directory defined in your Hadoop cluster for temporary files created by running programs. Typically, this directory can be found under the `yarn.app.mapreduce.am.staging-dir` property in the configuration files such as `yarn-site.xml` or `mapred-site.xml` of your distribution. |
4. Allocate proper memory volumes to the Map and the Reduce computations and the ApplicationMaster of YARN by selecting the Set memory check box in the Advanced settings view.

5. Select the Set Hadoop user check box and enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed.

6. Select the Use datanode hostname check box to allow the Job to access datanodes via their hostnames. This actually sets the dfs.client.use.datanode.hostname property to true. When connecting to a S3N filesystem, you must select this check box.

For further information about these parameters, see the documentation or contact the administrator of the Hadoop cluster to be used.

For further information about the Hadoop Map/Reduce framework, see the Map/Reduce tutorial in Apache's Hadoop documentation on http://hadoop.apache.org.

**Use kerberos authentication**

If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.

  Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the Force MapR ticket authentication check box and the Use Kerberos authentication check box clear, and then MapR should be able to automatically find that ticket on the fly.

In addition, since this component performs Map/Reduce computations, you also need to authenticate the related services such as the Job history server and the Resource manager or Jobtracker depending on your distribution in the corresponding field. These principals can be found in the configuration files of your distribution. For example, in a CDH4 distribution, the Resource manager principal is set in the yarn-site.xml file and the Job history principal in the mapred-site.xml file.

This check box is available depending on the Hadoop distribution you are connecting to.
### Use a keytab to authenticate

Select the **Use a keytab to authenticate** check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the **Principal** field and the access path to the keytab file itself in the **Keytab** field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is `user1` and the principal to be used is `guest`; in this situation, ensure that `user1` has the right to read the keytab file to be used.

### Hadoop user name

Enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed. Note that this field is available depending on the distribution you are using.

### JDBC property

Either **Built-in** or **Repository**:

- **Built-in**: you enter the connection information of the database to be used locally for this component only.

- **Repository**: you have already created the database connection and stored it in the **Repository**; therefore, you reuse it directly for the component configuration and the Job design. For further information about how to create a centralized database connection, see [Talend Studio User Guide](#).

Note that only the **General JDBC** connection stored in the **Repository** is supported.

### Connection

Enter the JDBC URL used to connect to the database where the source data is stored.

### User name and Password

Enter the authentication information used to connect to the source database.

To enter the password, click the `[...]` button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

If your password is stored in a file, select the **The password is stored in a file** check box and enter the path to that file in the **File path** field that is displayed.

- This file can be stored either in the machine where the Job is to be executed or in the HDFS system of the Hadoop cluster to be used.

- The password stored in this file must not contain `\n` (the newline escape) at the end, that is to say, you must not insert a new line at the end of the password even though this line is empty.
Note that this feature is available depending on the Sqoop version you are using.

### Driver JAR
In either the **Use Commandline** mode or the **Java API** mode, you must add the driver file of the database to be used to the *lib* folder of the Hadoop distribution you are using. For that purpose, use this **Driver JAR** table to add that driver file for the current Job you are designing.

### File format
Select a file format for the data to be transferred:
- `textfile`
- `sequencefile`
- `Avro file`
- `Parquet file`: the version of Sqoop must be 1.4.6.

### Specify Number of Mappers
Select this check box to indicate the number of map tasks (parallel processes) used to perform the data transfer.
If you do not want Sqoop to work in parallel, enter 1 in the displayed field.

### Compress
Select this check box to enable compression.

### Direct
Select this check box to use the import fast path.

### Exclude table
Select this check box and enter the name of the table(s) to be excluded from the import process.

### Print Log
Select this check box to activate the **Verbose** check box.

### Verbose
Select this check box to print more information while working, for example, the debugging information.

### Advanced settings

#### Define Hive mapping
Sqoop provides default configuration that maps most SQL types to appropriate Hive types. If you need to use your custom map to overwrite the default ones at runtime, select this check box and define the map(s) you want to use in the table that appears.

#### Use MySQL default delimiters
Select this check box to use MySQL's default delimiter set. This check box is available only to the Commandline mode.

#### Additional arguments
Complete this table to use additional arguments if needed be.
By adding additional arguments, you are able to perform multiple operations in one single transaction. For example, you can use `--hive-import` and `--hive-table` in the Commandline mode or `hive.import` and `hive.table.name` in the Java API mode to create Hive table and write data in at the runtime of the transaction writing data in HDFS. For further information about the available Sqoop arguments in the
Commandline mode and the Java API mode, respectively, see Additional arguments on page 3684.

| Hadoop properties | Talend Studio uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.  

- Note that if you are using the centrally stored metadata from the Repository, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from Repository to Built-in.  

For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on http://hadoop.apache.org/docs and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:

- Typically, the HDFS-related properties can be found in the hdfs-default.xml file of your distribution, such as http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml.

- Apache also provides a page to list the Hive-related properties: https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties. |

| Mapred job map memory mb and Mapred job reduce memory mb | You can tune the map and reduce computations by selecting the Set memory check box to set proper memory allocations for the computations to be performed by the Hadoop system.  

In that situation, you need to enter the values you need in the Mapred job map memory mb and the Mapred job reduce memory mb fields, respectively. By default, the values are both 1000 which are normally appropriate for running the computations.  

The memory parameters to be set are Map (in Mb), Reduce (in Mb) and ApplicationMaster (in Mb). These fields allow you to dynamically allocate memory to the map and the reduce computations and the ApplicationMaster of YARN. |

| Path separator in server | Leave the default value of the Path separator in server as it is, unless you have changed the separator used by your Hadoop distribution’s host machine for its PATH variable or in other words, that separator is not a colon (:). In that situation, you must change this value to the one you are using in that host. |

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |
### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>the error message generated by the component when an error occurs.</td>
</tr>
<tr>
<td></td>
<td>This is an After variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td>This variable functions only if the <strong>Die on error</strong> check box is cleared.</td>
</tr>
<tr>
<td></td>
<td>If the component has this check box.</td>
</tr>
<tr>
<td><strong>EXIT_CODE</strong></td>
<td>the exit code of the remote command.</td>
</tr>
<tr>
<td></td>
<td>This is an After variable and it returns an integer.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

### Usage

**Usage rule**

This component is used standalone. It respects the Sqoop prerequisites. You need necessary knowledge about Sqoop to use it.

We recommend using the Sqoop of version 1.4+ in order to benefit the full functions of these components.

For further information about Sqoop, see the Sqoop manual on: [http://sqoop.apache.org/docs/](http://sqoop.apache.org/docs/)

### Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with [Talend Studio](#). The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR's documentation, the library or libraries of a MapR client corresponding to each OS version can be found under `MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native`. For example, the library for Windows is `\lib\native\MapRClient.dll` in the MapR client jar file. For further information, see the following link from MapR: [http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr](http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-mapr).

    Without adding the specified library or libraries, you may encounter the following error: `no MapRClient in java.library.path`.

    - Set the `-Djava.library.path` argument, for example, in the **Job Run VM arguments** area of the **Run/Debug** view in the **[Preferences]** dialog box in the **Window** menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to
| **tSqoopImportAllTables** | make full use of the **Data viewer** to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using. |
|--------------------------|-------------------------------------------------------------------------------------------------|

**Limitation**

- If you have selected the **Use Commandline** mode, you need to use the host where Sqoop is installed to run the Job using this component.

- The preconditions required by Sqoop for using its `import-all-tables` tool must be satisfied. For further information, please see the manual of Sqoop.

**Connections**

- Outgoing links (from this component to another):
  - **Trigger**: Run if; On Subjob Ok; On Subjob Error.

- Incoming links (from one component to this one):
  - **Row**: Iterate;
  - **Trigger**: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error

- For further information regarding connections, see *Talend Studio User Guide*.

---

**Related scenarios**

No scenario is available for the Standard version of this component yet.
tSqoopMerge

Performs an incremental import that updates an older dataset with newer records. The file types of the newer and the older datasets must be the same.

tSqoopMerge reads two datasets in HDFS and combines them both using a merge class that is able to parse the datasets, with the newer records overwriting the older records.

**Note:**
Sqoop is typically installed in every Hadoop distribution. But if the Hadoop distribution you need to use have no Sqoop installed, you have to install one on your own and ensure to add the Sqoop command line to the PATH variable of that distribution. For further information about how to install Sqoop, see the documentation of Sqoop.

### tSqoopMerge Standard properties

These properties are used to configure tSqoopMerge running in the Standard Job framework.

The Standard tSqoopMerge component belongs to the Big Data and the File families.

The component in this framework is available when you are using one of the Talend solutions with Big Data.

#### Basic settings

<table>
<thead>
<tr>
<th>Mode</th>
<th>Select the mode in which Sqoop is called in a Job execution.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Commandline:</td>
<td>the Sqoop shell is used to call Sqoop.</td>
</tr>
<tr>
<td></td>
<td>In this mode, you have to deploy and run the Job in the</td>
</tr>
<tr>
<td></td>
<td>host where Sqoop is installed. Therefore, if you are a</td>
</tr>
<tr>
<td></td>
<td>subscription-based user, we recommend installing and</td>
</tr>
<tr>
<td></td>
<td>using a Jobserver provided by Talend in that host to run</td>
</tr>
<tr>
<td></td>
<td>the Job; if you are using one of the Talend solutions wi</td>
</tr>
<tr>
<td></td>
<td>th Big Data, you have to ensure that the Studio and the</td>
</tr>
<tr>
<td></td>
<td>Sqoop to be used are in the same machine.</td>
</tr>
<tr>
<td>Use Java API:</td>
<td>the Java API is used to call Sqoop. In this mode, the Job</td>
</tr>
<tr>
<td></td>
<td>can be run locally in the Studio but you need to configure</td>
</tr>
<tr>
<td></td>
<td>the connection to the Hadoop distribution to be used. Note</td>
</tr>
<tr>
<td></td>
<td>that JDK is required to execute the Job in the Java API</td>
</tr>
<tr>
<td></td>
<td>mode and the versions of the JDK kits installed in both</td>
</tr>
<tr>
<td></td>
<td>machines must be compatible with each other; for example,</td>
</tr>
<tr>
<td></td>
<td>the versions are the same or the JDK version of the Hadoop</td>
</tr>
<tr>
<td></td>
<td>machine is more recent.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hadoop properties</th>
<th>Either Built-in or Repository:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Built-in</strong>: you enter the configuration information of</td>
</tr>
<tr>
<td></td>
<td>the Hadoop distribution to be used locally for this</td>
</tr>
<tr>
<td></td>
<td>component only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: you have already created the Hadoop</td>
</tr>
<tr>
<td></td>
<td>connection and stored it in the Repository; therefore, you</td>
</tr>
<tr>
<td></td>
<td>reuse it directly for the component configuration and the</td>
</tr>
<tr>
<td></td>
<td>Job design. For further</td>
</tr>
</tbody>
</table>
### Distribution

Select the cluster you are using from the drop-down list. The options in the list vary depending on the component you are using. Among these options, the following ones requires specific configuration:

- If available in this **Distribution** drop-down list, the **Microsoft HD Insight** option allows you to use a Microsoft HD Insight cluster. For this purpose, you need to configure the connections to the HD Insight cluster and the Windows Azure Storage service of that cluster in the areas that are displayed. For detailed explanation about these parameters, search for configuring the connection manually on Talend Help Center ([https://help.talend.com](https://help.talend.com)).

- If you select **Amazon EMR**, you can find more details about how to configure an Amazon EMR cluster in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

- The **Custom** option allows you to connect to a cluster different from any of the distributions given in this list, that is to say, to connect to a cluster not officially supported by Talend.

1. Select **Import from existing version** to import an officially supported distribution as base and then add other required jar files which the base distribution does not provide.

2. Select **Import from zip** to import the configuration zip for the custom distribution to be used. This zip file should contain the libraries of the different Hadoop elements and the index file of these libraries.

   In **Talend Exchange**, members of **Talend community** have shared some ready-for-use configuration zip files which you can download from this **Hadoop configuration** list and directly use them in your connection accordingly. However, because of the ongoing evolution of the different Hadoop-related projects, you might not be able to find the configuration zip corresponding to your distribution from this list; then it is recommended to use the **Import from existing version** option to take an existing distribution as base to add the jars required by your distribution.

Note that custom versions are not officially supported by Talend. Talend and its community provide you with the opportunity to connect to custom versions from the Studio but cannot guarantee that the configuration of whichever version you choose will be easy, due to the wide range of different Hadoop distributions and versions that are available. As such, you should only attempt to set up such a connection if you have sufficient information about how to create a centralized Hadoop connection, see **Talend Open Studio for Big Data Getting Started Guide**.
Hadoop experience to handle any issues on your own.

**Note:**
In this dialog box, the active check box must be kept selected so as to import the jar files pertinent to the connection to be created between the custom distribution and this component.

For a step-by-step example about how to connect to a custom distribution and share this connection, see [Connecting to a custom Hadoop distribution](#) on page 1589.

<table>
<thead>
<tr>
<th><strong>Hadoop Version</strong></th>
<th>Select the version of the Hadoop distribution you are using. The available options vary depending on the component you are using.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NameNode URI</strong></td>
<td>Type in the URI of the Hadoop NameNode, the master node of a Hadoop system. For example, we assume that you have chosen a machine called <em>masternode</em> as the NameNode, then the location is <code>hdfs://masternode:portnumber</code>. If you are using WebHDFS, the location should be <code>webhdfs://masternode:portnumber</code>; if this WebHDFS is secured with SSL, the scheme should be <code>swebhdfs</code> and you need to use a <code>tLibraryLoad</code> in the Job to load the library required by the secured WebHDFS.</td>
</tr>
</tbody>
</table>
| **JobTracker Host** | Select this check box and in the displayed field, enter the location of the ResourceManager of your distribution. For example, `tal-qal14.taland.lan:8050`. This property is required when the query you want to use is executed in Windows and it is a Select query. For example, `SELECT your_column_name FROM your_table_name`

Then you can continue to set the following parameters depending on the configuration of the Hadoop cluster to be used (if you leave the check box of a parameter clear, then at runtime, the configuration about this parameter in the Hadoop cluster to be used will be ignored):

1. Select the **Set resourcemanager scheduler address** check box and enter the Scheduler address in the field that appears.

2. Select the **Set jobhistory address** check box and enter the location of the JobHistory server of the Hadoop cluster to be used. This allows the metrics information of the current Job to be stored in that JobHistory server.

3. Select the **Set staging directory** check box and enter this directory defined in your Hadoop cluster for temporary files created by running programs. Typically, this directory can be found under the `yarn.app.mapreduce.am.staging-dir` property in the

---
configuration files such as `yarn-site.xml` or `mapred-site.xml` of your distribution.

4. Allocate proper memory volumes to the Map and the Reduce computations and the ApplicationMaster of YARN by selecting the Set memory check box in the Advanced settings view.

5. Select the Set Hadoop user check box and enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed.

6. Select the Use datanode hostname check box to allow the Job to access datanodes via their hostnames. This actually sets the `dfs.client.use.datanode.hostname` property to true. When connecting to a S3N filesystem, you must select this check box.

For further information about these parameters, see the documentation or contact the administrator of the Hadoop cluster to be used.


### Use kerberos authentication

If you are accessing the Hadoop cluster running with Kerberos security, select this check box, then, enter the Kerberos principal name for the NameNode in the field displayed. This enables you to use your user name to authenticate against the credentials stored in Kerberos.

- If this cluster is a MapR cluster of the version 5.0.0 or later, you can set the MapR ticket authentication configuration in addition or as an alternative by following the explanation in Connecting to a security-enabled MapR on page 1652.

Keep in mind that this configuration generates a new MapR security ticket for the username defined in the Job in each execution. If you need to reuse an existing ticket issued for the same username, leave both the Force MapR ticket authentication check box and the Use Kerberos authentication check box clear, and then MapR should be able to automatically find that ticket on the fly.

In addition, since this component performs Map/Reduce computations, you also need to authenticate the related services such as the Job history server and the Resource manager or Jobtracker depending on your distribution in the corresponding field. These principals can be found in the configuration files of your distribution. For example, in a CDH4 distribution, the Resource manager principal is set in the `yarn-site.xml` file and the Job history principal in the `mapred-site.xml` file.
<table>
<thead>
<tr>
<th><strong>tSqoopMerge</strong></th>
<th>This check box is available depending on the Hadoop distribution you are connecting to.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use a keytab to authenticate</strong></td>
<td>Select the <strong>Use a keytab to authenticate</strong> check box to log into a Kerberos-enabled system using a given keytab file. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the <strong>Principal</strong> field and the access path to the keytab file itself in the <strong>Keytab</strong> field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver. Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is <strong>user1</strong> and the principal to be used is <strong>guest</strong>; in this situation, ensure that <strong>user1</strong> has the right to read the keytab file to be used.</td>
</tr>
<tr>
<td><strong>Hadoop user name</strong></td>
<td>Enter the user name under which you want to execute the Job. Since a file or a directory in Hadoop has its specific owner with appropriate read or write rights, this field allows you to execute the Job directly under the user name that has the appropriate rights to access the file or directory to be processed. Note that this field is available depending on the distribution you are using.</td>
</tr>
<tr>
<td><strong>Old data directory</strong></td>
<td>Enter the path to the older dataset to be merged.</td>
</tr>
<tr>
<td><strong>New data directory</strong></td>
<td>Enter the path to the newer dataset to be merged.</td>
</tr>
<tr>
<td><strong>Target directory</strong></td>
<td>Enter the directory where you need to put the output of the merging.</td>
</tr>
<tr>
<td><strong>Merge key</strong></td>
<td>Enter the name of the column used as the key of each record for the merging. This primary key must be unique.</td>
</tr>
<tr>
<td><strong>Need to generate the JAR file</strong></td>
<td>Select this check box to generate the merge jar file and the merge class required to parse the datasets to be merged. The default name of the jar file and the class is <strong>SqoopMerge_component_ID</strong>. This <strong>component_ID</strong> is the ID of the <strong>tSqoopMerge</strong> component that generates the jar file and the class, such as <strong>tSqoopMerge_1</strong>, or <strong>tSqoopMerge_2</strong>. As this jar file is generated from the source table of the imported data, selecting this check box displays the corresponding parameters to be set for connecting to that table.</td>
</tr>
</tbody>
</table>
| **JDBC property** | Either **Built-in** or **Repository**:  
  - **Built-in**: you enter the connection information of the database to be used locally for this component only.  
  - **Repository**: you have already created the database connection and stored it in the **Repository**; therefore, you reuse it directly for the component configuration and the Job design. For further |
information about how to create a centralized
database connection, see Talend Studio User Guide.

Note that only the General JDBC connection stored
in the Repository is supported.

| Connection                      | Enter the JDBC URL used to connect to the database
|                                | where the source data is stored.
| **User name and Password**      | Enter the authentication information used to connect to
|                                | the source database.
|                                | To enter the password, click the [...] button next to the
|                                | password field, and then in the pop-up dialog box enter
|                                | the password between double quotes and click OK to
|                                | save the settings.
| **Table Name**                  | Type in the name of the source table.
|                                | This name is used to name the generated jar file.
| **Driver JAR**                  | In either the Use Commandline mode or the Java API
|                                | mode, you must add the driver file of the database to be
|                                | used to the lib folder of the Hadoop distribution you are
|                                | using. For that purpose, use this Driver JAR table to add
|                                | that driver file for the current Job you are designing.
|                                | This driver jar is required only when you need to connect
to the database to be used to generate the merge jar
|                                | file; therefore this Driver JAR table is available only
|                                | when you have selected the Need to generate the JAR
|                                | file check box.
| **JAR file**                    | If a required merge class already exists and is available,
specify the access path to the jar file that contains that
class for reusing the class.
|                                | In this situation, you need to enter the name of the class
|                                | in the Class name field in the Advanced settings tab.
| **Print Log**                   | Select this check box to activate the Verbose check box.
| **Verbose**                     | Select this check box to print more information while
|                                | working, for example, the debugging information.

**Advanced settings**

| **Custom class name**          | Select this check box to display the Class name field
|                                | and enter the name of the merge class you need to use.
|                                | This check box must be clear if you use Generate the
|                                | JAR file in the Basic settings tab.
| **Additional arguments**        | Complete this table to use additional arguments if needs be.
|                                | By adding additional arguments, you are able to perform
|                                | multiple operations in one single transaction. For example,
you can use --hive-import and --hive-table in the Commandline mode or
hive.import and hive.table.name in the Java API mode to
create Hive table and write data in at the runtime of the transaction writing data in HDFS. For further information about the available Sqoop arguments in the Commandline mode and the Java API mode, respectively, see Additional arguments on page 3684.

<table>
<thead>
<tr>
<th>Hadoop properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Talend Studio</em> uses a default configuration for its engine to perform operations in a Hadoop distribution. If you need to use a custom configuration in a specific situation, complete this table with the property or properties to be customized. Then at runtime, the customized property or properties will override those default ones.</td>
</tr>
<tr>
<td>• Note that if you are using the centrally stored metadata from the <em>Repository</em>, this table automatically inherits the properties defined in that metadata and becomes uneditable unless you change the Property type from <em>Repository</em> to <em>Built-in</em>.</td>
</tr>
<tr>
<td>For further information about the properties required by Hadoop and its related systems such as HDFS and Hive, see the documentation of the Hadoop distribution you are using or see Apache’s Hadoop documentation on <a href="http://hadoop.apache.org/docs">http://hadoop.apache.org/docs</a> and then select the version of the documentation you want. For demonstration purposes, the links to some properties are listed below:</td>
</tr>
<tr>
<td>• Typically, the HDFS-related properties can be found in the <em>hdfs-default.xml</em> file of your distribution, such as <a href="http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml">http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml</a>.</td>
</tr>
<tr>
<td>• Apache also provides a page to list the Hive-related properties: <a href="https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties">https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties</a>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mapred job map memory mb and Mapred job reduce memory mb</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can tune the map and reduce computations by selecting the Set memory check box to set proper memory allocations for the computations to be performed by the Hadoop system.</td>
</tr>
<tr>
<td>In that situation, you need to enter the values you need in the Mapred job map memory mb and the Mapred job reduce memory mb fields, respectively. By default, the values are both 1000 which are normally appropriate for running the computations.</td>
</tr>
<tr>
<td>The memory parameters to be set are Map (in Mb), Reduce (in Mb) and ApplicationMaster (in Mb). These fields allow you to dynamically allocate memory to the map and the reduce computations and the ApplicationMaster of YARN.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Path separator in server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leave the default value of the Path separator in server as it is, unless you have changed the separator used by your Hadoop distribution’s host machine for its PATH variable or in other words, that separator is not a colon (:). In that situation, you must change this value to the one you are using in that host.</td>
</tr>
</tbody>
</table>
tStatCatcher Statistics

Select this check box to collect log data at the component level.

Global Variables

Global Variables

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

EXIT_CODE: the exit code of the remote command. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

This component is used standalone. It respects the Sqoop prerequisites. You need necessary knowledge about Sqoop to use it.

We recommend using the Sqoop of version 1.4+ in order to benefit the full functions of these components.

For further information about Sqoop, see the Sqoop manual on: http://sqoop.apache.org/docs/

Prerequisites

The Hadoop distribution must be properly installed, so as to guarantee the interaction with Talend Studio. The following list presents MapR related information for example.

- Ensure that you have installed the MapR client in the machine where the Studio is, and added the MapR client library to the PATH variable of that machine. According to MapR’s documentation, the library or libraries of a MapR client corresponding to each OS version can be found under MAPR_INSTALL\hadoop\hadoop-VERSION\lib\native. For example, the library for Windows is \lib\native\MapRClient.dll in the MapR client jar file. For further information, see the following link from MapR: http://www.mapr.com/blog/basic-notes-on-configuring-eclipse-as-a-hadoop-development-environment-for-map.

Without adding the specified library or libraries, you may encounter the following error: no MapRClient in java.library.path.

- Set the -Djava.library.path argument, for example, in the Job Run VM arguments area of the
Run/Debug view in the [Preferences] dialog box in the Window menu. This argument provides to the Studio the path to the native library of that MapR client. This allows the subscription-based users to make full use of the Data viewer to view locally in the Studio the data stored in MapR.

For further information about how to install a Hadoop distribution, see the manuals corresponding to the Hadoop distribution you are using.

**Limitation**

If you have selected the Use Commandline mode, you need to use the host where Sqoop is installed to run the Job using this component.

**Connections**

Outgoing links (from this component to another):

Trigger: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.

Incoming links (from one component to this one):

Row: Iterate;

Trigger: Run if; On Subjob Ok; On Subjob Error; On Component Ok; On Component Error.

For further information regarding connections, see Talend Studio User Guide.

---

**Scenario: Merging two datasets in HDFS**

This scenario applies only to a Talend solution with Big Data.

This scenario illustrates how to use **tSqoopMerge** to merge two datasets that are sequentially imported to HDFS from the same MySQL table, with modifications of a record in between.

The first dataset (the old one before the modifications) to be used in this scenario reads as follows:

<table>
<thead>
<tr>
<th>id, wage, mod_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 2000, 2008-06-26 04:25:59</td>
</tr>
<tr>
<td>1, 2300, 2011-06-12 05:29:45</td>
</tr>
<tr>
<td>3, 3000, 2010-05-02 15:34:05</td>
</tr>
</tbody>
</table>

The path to it in HDFS is /user/ychen/target_old.

The second dataset (the new one after the modifications) to be used reads as follows:

<table>
<thead>
<tr>
<th>id, wage, mod_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2300, 2011-06-12 05:29:45</td>
</tr>
<tr>
<td>3, 3000, 2010-05-02 15:34:05</td>
</tr>
</tbody>
</table>
The path to it in HDFS is /user/ychen/target_new.

These datasets were both imported by tSqoopImport. For a scenario about how to use tSqoopImport, see Scenario: Importing a MySQL table to HDFS on page 3705.

The Job in this scenario merges these two datasets with the newer record overwriting the older one. Before starting to replicate this scenario, ensure that you have appropriate rights and permissions to access the Hadoop distribution to be used. Then proceed as follows:

**Dropping the component**

**Procedure**

1. In the Integration perspective of the Studio, create an empty Job from the Job Designs node in the Repository tree view.
   For further information about how to create a Job, see Talend Studio User Guide.
2. Drop tSqoopMerge onto the workspace.
   In this scenario, the required jar file for the merge is not available, you then need to use tSqoopMerge to generate it at runtime from the source MySQL table.

**Configuring tSqoopMerge**

**Procedure**

1. Double-click tSqoopMerge to open its Component view.
2. In the **Mode** area, select **Use Java API**.

3. In the **Version** area, select the Hadoop distribution to be used and its version. If you cannot find from the list the distribution corresponding to yours, select **Custom** so as to connect to a Hadoop distribution not officially supported in the Studio.

   For a step-by-step example about how to use this **Custom** option, see Connecting to a custom Hadoop distribution on page 1589.

4. In the **NameNode URI** field, enter the location of the master node, the NameNode, of the distribution to be used. For example, `hdfs://talend-cdh4-namenode:8020`. If you are using WebHDFS, the location should be `webhdfs://masternode:portnumber`; if this WebHDFS is secured with SSL, the scheme should be `swebhdfs` and you need to use a `tLibraryLoad` in the Job to load the library required by the secured WebHDFS.

5. In the **Resource Manager** field, enter the location of the ResourceManager of your distribution.
6. If the distribution to be used requires Kerberos authentication, select the Use Kerberos authentication check box and complete the authentication details. Otherwise, leave this check box clear.

If you need to use a Kerberos keytab file to log in, select Use a keytab to authenticate. A keytab file contains pairs of Kerberos principals and encrypted keys. You need to enter the principal to be used in the Principal field and the access path to the keytab file itself in the Keytab field. This keytab file must be stored in the machine in which your Job actually runs, for example, on a Talend Jobserver.

Note that the user that executes a keytab-enabled Job is not necessarily the one a principal designates but must have the right to read the keytab file being used. For example, the user name you are using to execute a Job is user1 and the principal to be used is guest; in this situation, ensure that user1 has the right to read the keytab file to be used.

7. In the Old data directory and the New data directory fields, enter the path, or browse to the directory in HDFS where the older and the newer datasets are stored, respectively.

8. In the Target directory field, enter the path, or browse to the folder you need to store the merge result in.

9. In the Merge key field, enter the column to be used as the key for the merge. In this scenario, the column is id.

10. Select Need to generate the JAR file to display the connection parameters to the source database table.

11. In the Connection field, enter the URI of the MySQL database where the source table is stored. For example, jdbc:mysql://10.42.10.13/mysql.

12. In the Table Name field, enter the name of the source table. In this scenario, it is sqoopmerge.

13. In Username and Password, enter the authentication information.

14. Under the Driver JAR table, click the [+ ] button to add one row, then in this row, click the [...] button to display the drop-down list and select the jar file to be used from that list. In this scenario, it is mysql-connector-java-5.1.30-bin.jar.

   If the [...] button does not appear, click anywhere in this row to make it displayed.

15. If the field delimiter of the source table is not comma (,), you still need to specify the delimiter in the Additional Arguments table in the Advanced settings tab. The argument to be used is codegen.output.delimiters.field for the Use Java API mode or --fields-terminated-by for the Use Commandline mode.

**Executing the Job**

Then you can press F6 to run this Job.

During the execution, the jar file and the class for the merge are generated in the local machine.
Once done, you can verify the results in the target directory you have specified, in the web console of the Hadoop distribution used.

File: /user/ychen/merged/part-r-00000

Go back to dir listing
Advanced view/download options

If you need to obtain more details about the Job, it is recommended to use the web console of the Jobtracker provided by the Hadoop distribution you are using.

If you continue to import updated datasets to HDFS from the same source table, you can reuse the generated merge class to merge the datasets.
tSQSConnection

Opens a connection to Amazon Simple Queue Service that can then be reused by other SQS components.

**tSQSConnection Standard properties**

These properties are used to configure tSQSConnection running in the Standard Job framework.

The Standard tSQSConnection component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

| **Access Key** and **Secret Key** | Specify the access keys (the access key ID in the **Access Key** field and the secret access key in the **Secret Key** field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key).

To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| --- | --- |

<table>
<thead>
<tr>
<th><strong>Inherit credentials from AWS role</strong></th>
<th>Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</th>
</tr>
</thead>
</table>

| **Assume Role** | Select this check box and specify the values for the following parameters used to create a new assumed role session.

- **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.
- **Role session name**: an identifier for the assumed role session.
- **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.

For more information on assuming roles, see AssumeRole. |
| --- | --- |

| **Region** | Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (for example "us-east-1") in the list. For more information on AWS Regions, see AWS Regions and Endpoints. |
Advanced settings

| Config client | Select this check box and in the table displayed specify the client configuration parameters. Click the [+] button below the table to add as many rows as needed, each row for a client configuration parameter, and set the value of the following attributes for each parameter:
|               | • Client Parameter: click the cell and select a parameter from the drop-down list.  
|               | • Value: enter the value for the corresponding parameter. |

| STS Endpoint | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from. This check box is available only when the Assume role check box is selected. |

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

Global variables

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

Usage

| Usage rule | This component can be used as a standalone component of a Job or Subjob. |

Related scenarios

- Delivering messages to an Amazon SQS queue on page 3750
- Listing Amazon SQS queues in an AWS region on page 3766
- Retrieving messages from an Amazon SQS queue on page 3737
tSQSInput

Retrieves one or more messages, with a maximum limit of ten messages, from an Amazon SQS (Simple Queue Service) queue.

**tSQSInput Standard properties**

These properties are used to configure tSQSInput running in the Standard Job framework. The Standard tSQSInput component belongs to the Cloud family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Key and Secret Key</strong></td>
<td>Specify the access keys (the access key ID in the Access Key field and the secret access key in the Secret Key field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key). To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Inherit credentials from AWS role</strong></td>
<td>Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</td>
</tr>
</tbody>
</table>
| **Assume Role** | Select this check box and specify the values for the following parameters used to create a new assumed role session.  
  • **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.  
  • **Role session name**: an identifier for the assumed role session.  
  • **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.  
  For more information on assuming roles, see AssumeRole. |
| **Region** | Specify the AWS region by selecting a region name from the list or entering a region between double quotation
marks (for example 'us-east-1') in the list. For more information on AWS Regions, see [AWS Regions and Endpoints](https://docs.aws.amazon.com/AmazonSQS/latest/APIReference/).  

<table>
<thead>
<tr>
<th>Queue (Name or URL)</th>
<th>Specify the name or the URL of the queue from which messages will be retrieved.</th>
</tr>
</thead>
</table>
| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.  

The schema of this component is predefined with the following four String type message attributes:  

- **MessageId**: the ID of the message.  
- **ReceiptHandle**: the receipt handle of the message.  
- **MD5OfBody**: the MD5 digest of the message body.  
- **Body**: the message body.  

You can click the button next to `Edit schema` to view its schema and edit the schema by removing existing attributes or adding standard SQS attributes and customized user attributes. |
| Read standard attributes | Select this check box to retrieve standard Amazon SQS attributes along with each message. For more information on the Amazon SQS attributes, see [ReceiveMessage > Request Parameters](https://docs.aws.amazon.com/AmazonSQS/latest/APIReference/). |
| Read custom user attributes | Select this check box to retrieve customized user attributes along with each message. |
| Custom visibility timeout | Select this check box and in the field displayed specify the visibility timeout (in seconds) for the received messages. If not specified, the overall visibility timeout for the queue will be used for the received messages. For more information, see [Visibility Timeout](https://docs.aws.amazon.com/AmazonSQS/latest/APIReference/). |
| Custom wait time | Select this check box and in the field displayed specify the duration (in seconds) for which the call will wait for the message to arrive in the queue before returning. |
| Delete the messages while streaming | Select this check box to delete the message while retrieving it from the queue. |
| Read all messages from the queue | Select this check box to retrieve all messages from the queue.  

This check box is not available when the `Delete the messages while streaming` check box is cleared. |
| Max number of message to return per request | Specify the maximum number of messages to return per each request. Valid values are 1 to 10. |
| Die on error | Select the check box to stop the execution of the Job when an error occurs. |
Clear the check box to skip any error and continue the Job execution process.

**Advanced settings**

| Config client | Select this check box and in the table displayed specify the client configuration parameters. Click the [+] button below the table to add as many rows as needed, each row for a client configuration parameter, and set the value of the following attributes for each parameter:
|               | • **Client Parameter**: click the cell and select a parameter from the drop-down list.
|               | • **Value**: enter the value for the corresponding parameter.

| STS Endpoint | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from. This check box is available only when the Assume role check box is selected.

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global variables**

| NB_LINE | The number of rows processed. This is an After variable and it returns an integer.

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is selected.

**Usage**

| Usage rule | This component is usually used as a start component of a Job or Subjob and it always needs an output link.

**Retrieving messages from an Amazon SQS queue**

Here’s an example of using Talend components to retrieve all messages from an existing Amazon SQS (Simple Queue Service) queue and then delete the empty queue.

**Creating a Job for retrieving Amazon SQS messages**

Create a Job to open a connection to access Amazon SQS, then retrieve all messages from an existing Amazon SQS queue, finally delete the queue.
Prerequisites: You must deliver the following message creator attribute and message body data to the Amazon SQS queue named talend in order to replicate this scenario.

#Creator;Body
Talend DOC Team;the 1st message from Talend DOC team
Talend DEV Team;the 2nd message from Talend DEV team
Talend QA Team;the 3rd message from Talend QA team

Procedure

1. Create a new Job and add a tSQSConnection component, a tSQSInput component, a tLogRow component, and a tSQSQueueDelete component by typing their names in the design workspace or dropping them from the Palette.
2. Link the tSQSInput component to the tLogRow component using a Row > Main connection.
3. Link the tSQSConnection component to the tSQSInput component using a Trigger > OnSubjobOk connection.
4. Link the tSQSInput component to the tSQSQueueDelete component using a Trigger > OnSubjobOk connection.

Opening a connection to access Amazon SQS

Configure the tSQSConnection component to open a connection to access Amazon SQS.

Procedure

1. Double-click the tSQSConnection component to open its Basic settings view.
2. In the **Access Key** and **Secret Key** fields, specify the authentication information required to access Amazon SQS.

3. Select an AWS region available for Amazon SQS from the **Region** drop-down list. In this example, it is **Asia Pacific (Tokyo)**.

### Retrieving messages from an Amazon SQS queue

Configure the **tSQSInput** component and the **LogRow** component to retrieve all messages from an existing Amazon SQS queue and display the retrieved messages on the console of Talend Studio.

**Procedure**

1. Double-click the **tSQSInput** component to open its **Basic settings** view.

2. Specify the connection details required to access Amazon SQS. In this example, select the **Use an existing connection** check box and from the **Component List** drop-down list displayed, select the connection component to reuse its connection details you have already defined.

3. In the **Queue (Name or URL)** field, enter the name for the queue from which the messages will be retrieved. In this example, it is **talend**.

4. Click the button next to **Edit schema** to open the schema dialog box.
You can see the schema of the `tSQSInput` component is predefined with four columns of String type: `MessageId`, `ReceiptHandle`, `MD5OfBody` and `Body`.

5. Click the button to add five columns of String type, including four Amazon SQS attributes `ApproximateFirstReceiveTimestamp`, `ApproximateReceiveCount`, `SenderId` and `SentTimestamp` and one customized attribute `Creator`.

6. Click OK to save the changes and close the schema dialog box.

7. Select the Read standard attributes and Read custom user attributes check boxes to retrieve the value of four Amazon SQS attributes and one customized user attribute along with each message.

8. Double-click the `tLogRow` component to open its Basic settings view, and then select Table (print values in cells of a table) in the Mode area for better readability of the result.

### Deleting an Amazon SQS queue

Configure the `tSQSQueueDelete` component to create an Amazon SQS queue.

**Procedure**

1. Double-click the `tSQSQueueDelete` component to open its Basic settings view.
2. Specify the connection details required to access Amazon SQS. In this example, select the **Use an existing connection** check box and from the **Component List** drop-down list displayed, select the connection component to reuse its connection details you have already defined.

3. In the **Queue (Name or URL)** field, enter the name for the queue to be deleted. In this example, it is `talend`.

### Executing the Job to retrieve Amazon SQS messages

After setting up the Job and configuring the components used in the Job for retrieving Amazon SQS messages, you can then execute the Job and verify the Job execution result.

#### Procedure

1. Press **Ctrl + S** to save the Job and then **F6** to execute the Job.

As shown above, all three messages and their attributes are retrieved and displayed on the console.

2. View the queue details on the AWS console to validate the Job execution result.
You can see the queue talend has already been deleted and is not on the AWS console any more.
**tSQSMessageChangeVisibility**

Changes the visibility timeout of a specified message in an Amazon SQS (Simple Queue Service) queue.

**tSQSMessageChangeVisibility Standard properties**

These properties are used to configure tSQSMessageChangeVisibility running in the Standard Job framework.

The Standard tSQSMessageChangeVisibility component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
</table>
| **Access Key and Secret Key** | Specify the access keys (the access key ID in the Access Key field and the secret access key in the Secret Key field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key).  
To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Inherit credentials from AWS role** | Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances. |
| **Assume Role** | Select this check box and specify the values for the following parameters used to create a new assumed role session.  
• **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.  
• **Role session name**: an identifier for the assumed role session.  
• **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.  
For more information on assuming roles, see AssumeRole. |
<table>
<thead>
<tr>
<th>Region</th>
<th>Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (for example 'us-east-1') in the list. For more information on AWS Regions, see AWS Regions and Endpoints.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue (Name or URL)</td>
<td>Specify the name or the URL of the queue whose message’s visibility timeout will be changed.</td>
</tr>
<tr>
<td>Receipt Handle</td>
<td>Specify the receipt handle associated with the message whose visibility timeout will be changed.</td>
</tr>
<tr>
<td>Visibility Timeout in Seconds</td>
<td>Enter the new visibility timeout value in seconds for the specified message. Valid values are 0 to 43200.</td>
</tr>
<tr>
<td>Die on error</td>
<td>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any error and continue the Job execution process.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| Config client | Select this check box and in the table displayed specify the client configuration parameters. Click the (+) button below the table to add as many rows as needed, each row for a client configuration parameter, and set the value of the following attributes for each parameter:  
- **Client Parameter**: click the cell and select a parameter from the drop-down list.  
- **Value**: enter the value for the corresponding parameter. |
| STS Endpoint | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.  
This check box is available only when the Assume role check box is selected. |
| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global variables**

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is selected. |

**Usage**

| Usage rule | This component can be used as a standalone component of a Job or Subjob. |
Related scenario

No scenario is available for this component yet.
tSQSMessageDelete

Deletes a specified message from an Amazon SQS (Simple Queue Service) queue.

**tSQSMessageDelete Standard properties**

These properties are used to configure tSQSMessageDelete running in the Standard Job framework.

The Standard tSQSMessageDelete component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
</table>
| Access Key and Secret Key  | Specify the access keys (the access key ID in the Access Key field and the secret access key in the Secret Key field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key).
To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Inherit credentials from AWS role | Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances. |
| Assume Role               | Select this check box and specify the values for the following parameters used to create a new assumed role session.
- **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.
- **Role session name**: an identifier for the assumed role session.
- **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.
For more information on assuming roles, see AssumeRole. |
| Region                    | Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (for example 'us-east-1') in the list. For more |
information on AWS Regions, see [AWS Regions and Endpoints](#).

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue (Name or URL)</td>
<td>Specify the name or the URL of the queue from which a message will be deleted.</td>
</tr>
<tr>
<td>Receipt Handle</td>
<td>Specify the receipt handle associated with the message to be deleted.</td>
</tr>
<tr>
<td>Die on error</td>
<td>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any error and continue the Job execution process.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
</table>
| Config client            | Select this check box and in the table displayed specify the client configuration parameters. Click the \[+] button below the table to add as many rows as needed, each row for a client configuration parameter, and set the value of the following attributes for each parameter:  
  - **Client Parameter**: click the cell and select a parameter from the drop-down list.  
  - **Value**: enter the value for the corresponding parameter. |
|STS Endpoint              | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.  
  This check box is available only when the **Assume role** check box is selected. |
| tStatCatcher Statistics  | Select this check box to gather the Job processing metadata at the Job level as well as at each component level.                             |

**Global variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is selected.</td>
</tr>
</tbody>
</table>

**Usage**

| Usage rule                | This component can be used as a standalone component of a Job or Subjob.                                                                     |

**Related scenario**

No scenario is available for this component yet.
**tSQSOutput**

Delivers one or more messages to an Amazon SQS (Simple Queue Service) queue.

**tSQSOutput Standard properties**

These properties are used to configure tSQSOutput running in the Standard Job framework.

The Standard tSQSOutput component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Key</strong> and <strong>Secret Key</strong></td>
<td>Specify the access keys (the access key ID in the Access Key field and the secret access key in the Secret Key field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key). To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Inherit credentials from AWS role</strong></td>
<td>Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</td>
</tr>
</tbody>
</table>
| **Assume Role** | Select this check box and specify the values for the following parameters used to create a new assumed role session.  
  - **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.  
  - **Role session name**: an identifier for the assumed role session.  
  - **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.  
  For more information on assuming roles, see AssumeRole. |
| **Region** | Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (for example 'us-east-1') in the list. For more |
information on AWS Regions, see [AWS Regions and Endpoints](#).

<table>
<thead>
<tr>
<th>Queue (Name or URL)</th>
<th>Specify the name or the URL of the queue to which messages will be delivered. The name of a FIFO queue must end with the <code>.fifo</code> suffix.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields. The schema of this component is predefined with the following columns. You can click the <code>[...&gt;</code> button next to <strong>Edit schema</strong> to view its schema and edit the schema by adding some customized attributes for the messages to be delivered.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Body</strong>: the body of the message to be delivered.</td>
</tr>
<tr>
<td></td>
<td>• <strong>MessageGroupId</strong>: the specific message group that the message belongs to. Available only when the <strong>Use FIFO queues (first-in-first-out)</strong> check box is selected.</td>
</tr>
<tr>
<td></td>
<td>• <strong>MessageDeduplicationId</strong>: the token used for deduplication of the message. Available only when the <strong>Use FIFO queues (first-in-first-out)</strong> check box is selected and the <strong>Use Content Based Deduplication</strong> check box is cleared.</td>
</tr>
<tr>
<td></td>
<td>For more information on each predefined column, see <a href="#">SendMessage</a>.</td>
</tr>
<tr>
<td><strong>Use batch mode</strong></td>
<td>Select this check box to deliver messages to the queue using batch request. In the <strong>Batch size</strong> field displayed, enter the number of messages to be delivered in each batch. Note that the maximum number of messages delivered in each batch is 10.</td>
</tr>
<tr>
<td><strong>Use FIFO (first-in-first-out) queues</strong></td>
<td>Select this check box to deliver messages to a FIFO queue. For more information on the FIFO queue, see <a href="#">FIFO (First-In-First-Out) Queues</a>.</td>
</tr>
<tr>
<td><strong>Use Content Based Deduplication</strong></td>
<td>Select this check box to enable content-based deduplication. With this check box selected, Amazon SQS will use a SHA-256 hash to generate the message deduplication ID using the body of the message. This property is available only when the <strong>Use FIFO (first-in-first-out) queues</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any error and continue the Job execution process.</td>
</tr>
</tbody>
</table>
### Advanced settings

| Config client | Select this check box and in the table displayed specify the client configuration parameters. Click the [+] button below the table to add as many rows as needed, each row for a client configuration parameter, and set the value of the following attributes for each parameter:
| Client Parameter: click the cell and select a parameter from the drop-down list.
| Value: enter the value for the corresponding parameter. |

| STS Endpoint | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.
| This check box is available only when the Assume role check box is selected. |

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

### Global variables

| NB_LINE | The number of rows processed. This is an After variable and it returns an integer. |
| NB_LINE_SUCCESS | The number of rows successfully processed. This is an After variable and it returns an integer. |
| NB_LINE_REJECTED | The number of rows rejected. This is an After variable and it returns an integer. |
| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is selected. |

### Usage

| Usage rule | This component is usually used as an end component of a Job or Subjob and it always needs an input link. |

### Delivering messages to an Amazon SQS queue

Here’s an example of using Talend components to create an Amazon SQS (Simple Queue Service) queue, then deliver a couple of messages to the queue, finally get and display the attributes of the queue on the console.

The input messages in this example are as follows, and each message contains the message creator attribute and the message body information.

#Creator;Body
Creating a Job for delivering Amazon SQS messages

Create a Job to open a connection to access Amazon SQS, then create an Amazon SQS queue and deliver a couple of messages to the queue, finally get and display the attributes of the queue on the console.

Procedure

1. Create a new Job and add a `tSQSConnection` component, a `tSQSQueueCreate` component, a `tFixedSizeInput` component, a `tSQSOutput` component, a `tSQSQueueAttributes` component, and a `tLogRow` component by typing their names in the design workspace or dropping them from the Palette.

2. Link the `tFixedSizeInput` component to the `tSQSOutput` component using a Row > Main connection.

3. Do the same to link the `tSQSQueueAttributes` component to the `tLogRow` component.

4. Link the `tSQSConnection` component to the `tSQSQueueCreate` component using a Trigger > OnSubjobOk connection.

5. Do the same to link the `tSQSQueueCreate` component to the `tFixedSizeInput` component and the `tFixedSizeInput` component to the `tSQSQueueAttributes` component.

Opening a connection to access Amazon SQS

Configure the `tSQSConnection` component to open a connection to access Amazon SQS.
Procedure

1. Double-click the **tSQSConnection** component to open its **Basic settings** view.

   ![tSQSConnection](image)

   - **Access Key**: BKIOH3CC7E45S6
   - **Secret Key**: ************
   - **Inherit credentials from AWS role**: unchecked
   - **Assume Role**: unchecked
   - **Region**: Asia Pacific (Tokyo)

2. In the **Access Key** and **Secret Key** fields, specify the authentication information required to access Amazon SQS.

3. Select an AWS region available for Amazon SQS from the **Region** drop-down list. In this example, it is Asia Pacific (Tokyo).

Creating an Amazon SQS queue

Configure the **tSQSQueueCreate** component to create an Amazon SQS queue.

Procedure

1. Double-click the **tSQSQueueCreate** component to open its **Basic settings** view.

   ![tSQSQueueCreate](image)

   - **Connection**: Use an existing connection
   - **Queue (Name or URL)**: talend
   - **Queue Attributes**:
     - **MessageRetentionPeriod**: 86400

2. Specify the connection details required to access Amazon SQS. In this example, select the **Use an existing connection** check box and from the **Component List** drop-down list displayed, select the connection component to reuse its connection details you have already defined.

3. In the **Queue (Name or URL)** field, specify the name for the queue to be created. In this example, it is talend.

4. If needed, specify the attributes for the queue to be created in the **Queue Attributes** table. In this example, the **MessageRetentionPeriod** attribute is added and its value is set to 86400, which means Amazon SQS will retain the messages delivered to the queue for 86400 seconds (1 day) instead of the default 345600 seconds (4 day).
Delivering messages to the Amazon SQS queue

Configure the tFixedFlowInput component and the tSQSOutput component to deliver messages to the new empty Amazon SQS queue.

Procedure

1. Double-click the tSQSOutput component to open its Basic settings view.

2. Specify the connection details required to access Amazon SQS. In this example, select the Use an existing connection check box and from the Component List drop-down list displayed, select the connection component to reuse its connection details you have already defined.

3. In the Queue (Name or URL) field, enter the name for the queue to which the messages will be delivered. In this example, it is talend.

4. Click the button next to Edit schema to open the schema dialog box.

You can see the schema of the tSQSOutput component is predefined with a single String column Body that stores the body of each message to be delivered.

5. Click the button in the right panel to add one column Creator of String type, which will hold the customized creator attribute value of each message.

6. Click the button to copy all columns from the output schema to the input schema. In the pop-up dialog box, click OK to accept the changes.

7. Click OK to close the schema dialog box.

8. Double-click the tFixedFlowInput component to open its Basic settings view.
9. In the Mode area, select Use Inline Content(delimited file). Then in the Content field displayed, enter the creator attribute and body data of the messages to be delivered to Amazon SQS. In this example, it is:

- Talend DOC Team; the 1st message from Talend DOC team
- Talend DEV Team; the 2nd message from Talend DEV team
- Talend QA Team; the 3rd message from Talend QA team

### Getting attributes of the Amazon SQS queue

Configure the tSQSQueueAttributes component and the tLogRow component to get and display the attributes of the Amazon SQS queue on the console.

**Procedure**

1. Double-click the tSQSQueueAttributes component to open its Basic settings view.
2. Specify the connection details required to access Amazon SQS. In this example, select the Use an existing connection check box and from the Component List drop-down list displayed, select the connection component to reuse its connection details you have already defined.
3. In the Queue (Name or URL) field, enter the name for the queue whose attributes will be retrieved. In this example, it is talend.
4. Double-click the tLogRow component to open its Basic settings view.
5. In the Mode area, select Vertical to display each attribute and its value in a row.

### Executing the Job to deliver Amazon SQS messages

After setting up the Job and configuring the components used in the Job for delivering Amazon SQS messages, you can then execute the Job and verify the Job execution result.

**Procedure**

1. Press Ctrl + S to save the Job and then F6 to execute the Job.
As shown above, the Job is executed successfully and the attributes of the queue is displayed on the console.

2. View the queue details on the AWS console to validate the Job execution result.

![AWS Console](image)

You can see the queue talend is created successfully with the retention period for one day, and three messages are now available in the queue.

3. Select the queue talend on the AWS console and then click **Queue Actions > View/Delete Messages** to view messages in the queue.
You can see the three messages are now available in the queue.

4. Click **More Details** for one of the messages to view the detailed information of the message.

You can see the customized Creator attribute value for the message is listed on the **Message Attributes** tab.
**tSQSQueueAttributes**

Gets attributes for a specified Amazon SQS (Simple Queue Service) queue.

**tSQSQueueAttributes Standard properties**

These properties are used to configure tSQSQueueAttributes running in the Standard Job framework.

The Standard tSQSQueueAttributes component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use an existing connection</strong></td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td><strong>Access Key and Secret Key</strong></td>
<td>Specify the access keys (the access key ID in the <strong>Access Key</strong> field and the secret access key in the <strong>Secret Key</strong> field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key). To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Inherit credentials from AWS role</strong></td>
<td>Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</td>
</tr>
<tr>
<td><strong>Assume Role</strong></td>
<td>Select this check box and specify the values for the following parameters used to create a new assumed role session.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Role ARN</strong>: the Amazon Resource Name (ARN) of the role to assume.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Role session name</strong>: an identifier for the assumed role session.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Session duration (minutes)</strong>: the duration (in minutes) for which we want to have the assumed role session to be active.</td>
</tr>
<tr>
<td></td>
<td>For more information on assuming roles, see AssumeRole.</td>
</tr>
<tr>
<td><strong>Region</strong></td>
<td>Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (for example 'us-east-1') in the list. For more</td>
</tr>
</tbody>
</table>
**Queue (Name or URL)**
Specify the name or the URL of the queue whose attribute information will be retrieved.

**Schema and Edit schema**
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

The schema of this component is read-only and predefined with a couple of columns that describe the attributes for the specified queue. You can click the [...] button next to `Edit schema` to view its schema. For more information on each attribute, see GetQueueAttributes.

**Die on error**
Select the check box to stop the execution of the Job when an error occurs.
Clear the check box to skip any error and continue the Job execution process.

---

**Advanced settings**

**Config client**
Select this check box and in the table displayed specify the client configuration parameters. Click the [+] button below the table to add as many rows as needed, each row for a client configuration parameter, and set the value of the following attributes for each parameter:

- **Client Parameter**: click the cell and select a parameter from the drop-down list.
- **Value**: enter the value for the corresponding parameter.

**STS Endpoint**
Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.

This check box is available only when the Assume role check box is selected.

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

---

**Global variables**

**ERROR_MESSAGE**
The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is selected.
Usage

| Usage rule | This component is usually used as a start component of a Job or Subjob and it always needs an output link. |

Related scenario

Delivering messages to an Amazon SQS queue on page 3750
tSQSQueueCreate

Creates a new Amazon SQS (Simple Queue Service) queue.

**tSQSQueueCreate Standard properties**

These properties are used to configure tSQSQueueCreate running in the Standard Job framework.

The Standard tSQSQueueCreate component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Use an existing connection</strong></th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Key</strong> and <strong>Secret Key</strong></td>
<td>Specify the access keys (the access key ID in the Access Key field and the secret access key in the Secret Key field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key). To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Inherit credentials from AWS role</strong></td>
<td>Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</td>
</tr>
</tbody>
</table>
| **Assume Role** | Select this check box and specify the values for the following parameters used to create a new assumed role session.  
- **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.  
- **Role session name**: an identifier for the assumed role session.  
- **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.  
For more information on assuming roles, see AssumeRole. |
| **Region** | Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (for example ‘us-east-1’) in the list. For more |
information on AWS Regions, see [AWS Regions and Endpoints](#).

<table>
<thead>
<tr>
<th><strong>Queue (Name or URL)</strong></th>
<th>Specify the name of the queue to be created.</th>
</tr>
</thead>
</table>

**Queue Attributes**

Specify the attributes for the queue to be created. Click the button below the table to add as many rows as needed, each row for a queue attribute, and set the value of the following parameters for each attribute:

- **Attribute**: click the cell and select the name of the attribute from the drop-down list.
- **Value**: enter the value for the corresponding attribute.

If no value is defined for any attribute, the queue will have the default value for that attribute.

For more information on the queue attributes, see [CreateQueue > Request Parameters](#).

<table>
<thead>
<tr>
<th><strong>Die on error</strong></th>
<th>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any error and continue the Job execution process.</th>
</tr>
</thead>
</table>

### Advanced settings

**Config client**

Select this check box and in the table displayed specify the client configuration parameters. Click the + button below the table to add as many rows as needed, each row for a client configuration parameter, and set the value of the following attributes for each parameter:

- **Client Parameter**: click the cell and select a parameter from the drop-down list.
- **Value**: enter the value for the corresponding parameter.

**STS Endpoint**

Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.

This check box is available only when the **Assume role** check box is selected.

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global variables

<table>
<thead>
<tr>
<th><strong>ERROR_MESSAGE</strong></th>
<th>The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is selected.</th>
</tr>
</thead>
</table>
Usage

| Usage rule | This component can be used as a standalone component of a Job or Subjob. |

Related scenario

Delivering messages to an Amazon SQS queue on page 3750
tSQSQueueDelete

Deletes an Amazon SQS (Simple Queue Service) queue.

**tSQSQueueDelete Standard properties**

These properties are used to configure tSQSQueueDelete running in the Standard Job framework. The Standard tSQSQueueDelete component belongs to the Cloud family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Use an existing connection</strong></th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Key and Secret Key</strong></td>
<td>Specify the access keys (the access key ID in the Access Key field and the secret access key in the Secret Key field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key). To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Inherit credentials from AWS role</strong></td>
<td>Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</td>
</tr>
</tbody>
</table>
| **Assume Role** | Select this check box and specify the values for the following parameters used to create a new assumed role session.  
• **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.  
• **Role session name**: an identifier for the assumed role session.  
• **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.  
For more information on assuming roles, see AssumeRole. |
| **Region** | Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (for example 'us-east-1') in the list. For more |
information on AWS Regions, see AWS Regions and Endpoints.

<table>
<thead>
<tr>
<th>Queue (Name or URL)</th>
<th>Specify the name or the URL of the queue to be deleted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die on error</td>
<td>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any error and continue the Job execution process.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| Config client       | Select this check box and in the table displayed specify the client configuration parameters. Click the [+] button below the table to add as many rows as needed, each row for a client configuration parameter, and set the value of the following attributes for each parameter:  
|                    | • **Client Parameter**: click the cell and select a parameter from the drop-down list.  
|                    | • **Value**: enter the value for the corresponding parameter. |
| STS Endpoint       | Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from. This check box is available only when the **Assume role** check box is selected. |
| tStatCatcherStatistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Global variables**

| ERROR_MESSAGE | The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is selected. |

**Usage**

| Usage rule | This component can be used as a standalone component of a Job or Subjob. |

**Related scenario**

Retrieving messages from an Amazon SQS queue on page 3737
tSQSQueueList

Iterates and lists the URL of Amazon SQS (Simple Queue Service) queues in a specified region.

**tSQSQueueList Standard properties**

These properties are used to configure tSQSQueueList running in the Standard Job framework.

The Standard tSQSQueueList component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Key</strong> and <strong>Secret Key</strong></td>
<td>Specify the access keys (the access key ID in the <strong>Access Key</strong> field and the secret access key in the <strong>Secret Key</strong> field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key). To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Inherit credentials from AWS role</strong></td>
<td>Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.</td>
</tr>
</tbody>
</table>
| **Assume Role** | Select this check box and specify the values for the following parameters used to create a new assumed role session.  
  - **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.  
  - **Role session name**: an identifier for the assumed role session.  
  - **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active. For more information on assuming roles, see AssumeRole. |
| **Region** | Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (for example ‘us-east-1’) in the list. For more
Advanced settings

**Config client**
Select this check box and in the table displayed specify the client configuration parameters. Click the [+] button below the table to add as many rows as needed, each row for a client configuration parameter, and set the value of the following attributes for each parameter:
- **Client Parameter**: click the cell and select a parameter from the drop-down list.
- **Value**: enter the value for the corresponding parameter.

**STS Endpoint**
Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from.
This check box is available only when the **Assume role** check box is selected.

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Global variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENT_QUEUE_NAME</td>
<td>The URL of the queue being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>NB_QUEUE</td>
<td>The number of queues processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

**Usage rule**
This component is usually used as a start component of a Job or Subjob and it always needs an output link.

**Listing Amazon SQS queues in an AWS region**

Here's an example of using Talend components to iterate and list the URLs of all Amazon SQS (Simple Queue Service) queues in an AWS region.

**Creating a Job for listing Amazon SQS queues**

Create a Job to open a connection to access Amazon SQS, then iterate and list the URLs of all Amazon SQS queues in a specified region, finally display the total number of the queues in the region.
Prerequisites: You must create several Amazon SQS queues in an AWS region in order to replicate this scenario. In this example, four queues `talend`, `talend-dev`, `talend-doc` and `talend-qa` have been created in the region `Asia Pacific (Tokyo)` as shown in below figure.

![SQS Queues](image)

Procedure

1. Create a new Job and add a `tSQSConnection` component, a `tSQSQueueList` component, a `tIterateToFlow` component, a `tLogRow` component, and a `tJava` component by typing their names in the design workspace or dropping them from the Palette.
2. Link the `tSQSQueueList` component to the `tIterateToFlow` component using a `Row > Iterate` connection.
3. Link the `tIterateToFlow` component to the `tLogRow` component using a `Row > Main` connection.
4. Link the `tSQSConnection` component to the `tSQSQueueList` component using a `Trigger > OnSubjobOk` connection.
5. Link the `tSQSQueueList` component to the `tJava` component using a `Trigger > OnSubjobOk` connection.
Opening a connection to access Amazon SQS

Configure the **tSQSConnection** component to open a connection to access Amazon SQS.

**Procedure**

1. Double-click the **tSQSConnection** component to open its **Basic settings** view.

2. In the **Access Key** and **Secret Key** fields, specify the authentication information required to access Amazon SQS.
3. Select an AWS region available for Amazon SQS from the **Region** drop-down list. In this example, it is Asia Pacific (Tokyo).

Listing all Amazon SQS queues in an AWS region

Configure the **tSQSQueueList** component, the **tIterateToFlow** component, the **tLogRow** component, and the **tJava** component to list the URLs of all Amazon SQS queues in an AWS region and display the total number of the queues in the region on the console of Talend Studio.

**Procedure**

1. Double-click the **tSQSQueueList** component to open its **Basic settings** view.

2. Specify the connection details required to access Amazon SQS. In this example, select the **Use an existing connection** check box and from the **Component List** drop-down list displayed, select the connection component to reuse its connection details you have already defined.
3. Double-click the **tIterateToFlow** component to open its **Basic settings** view.
4. Click the button next to Edit schema to open the schema dialog box.

5. Click the button to add one String type column CurrentQueueURL that will hold the URLs of the queues to be listed. When done, click OK to close the dialog box.

6. In the Mapping table, set the value for the CurrentQueueURL column. In this example, the value is set to ((String)globalMap.get("tSQSQueueList_1_CURRENT_QUEUE_NAME")), which is the value of the global variable CURRENT_QUEUE_NAME for the tSQSQueueList component.

Note that you can fill the value by pressing Ctrl + Space to access the global variables list and then selecting tSQSQueueList_1_CURRENT_QUEUE_NAME from the list.

7. Double-click the tLogRow component to open its Basic settings view, and then select Table (print values in cells of a table) in the Mode area for better readability of the result.

8. Double-click the tJava component to open its Basic settings view.
9. In the **Code** field, enter the following code to display the total number of the queues in the region.

```java
System.out.println("The number of queues: " + ((Integer)globalMap.get("tSQSQueueList_1_NB_QUEUE"));
```

**Executing the Job to list Amazon SQS queues**

After setting up the Job and configuring the components used in the Job for listing Amazon SQS queues, you can then execute the Job and verify the Job execution result.

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to execute the Job.

As shown above, the URLs of all queues in the **Asia Pacific (Tokyo)** region and the total number of the queues are displayed on the console.
tSQSQueuePurge

Purges messages in an Amazon SQS (Simple Queue Service) queue.

**tSQSQueuePurge Standard properties**

These properties are used to configure tSQSQueuePurge running in the Standard Job framework.

The Standard tSQSQueuePurge component belongs to the Cloud family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
</table>
| **Access Key** and **Secret Key** | Specify the access keys (the access key ID in the Access Key field and the secret access key in the Secret Key field) required to access the Amazon Web Services. For more information on AWS access keys, see Access keys (access key ID and secret access key).

To enter the secret key, click the [...] button next to the secret key field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Inherit credentials from AWS role** | Select this check box to leverage the instance profile credentials. These credentials can be used on Amazon EC2 instances, and are delivered through the Amazon EC2 metadata service. To use this option, your Job must be running within Amazon EC2 or other services that can leverage IAM Roles for access to resources. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances. |
| **Assume Role** | Select this check box and specify the values for the following parameters used to create a new assumed role session.

- **Role ARN**: the Amazon Resource Name (ARN) of the role to assume.
- **Role session name**: an identifier for the assumed role session.
- **Session duration (minutes)**: the duration (in minutes) for which we want to have the assumed role session to be active.

For more information on assuming roles, see AssumeRole. |
| **Region** | Specify the AWS region by selecting a region name from the list or entering a region between double quotation marks (for example ‘us-east-1’) in the list. For more |
Queue (Name or URL)  Specify the name or the URL of the queue in which messages will be purged.

Die on error  Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any error and continue the Job execution process.

Advanced settings

Config client  Select this check box and in the table displayed specify the client configuration parameters. Click the [+] button below the table to add as many rows as needed, each row for a client configuration parameter, and set the value of the following attributes for each parameter:

- **Client Parameter**: click the cell and select a parameter from the drop-down list.
- **Value**: enter the value for the corresponding parameter.

STS Endpoint  Select this check box and in the field displayed, specify the AWS Security Token Service endpoint where session credentials are retrieved from. This check box is available only when the Assume role check box is selected.

tStatCatcher Statistics  Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

Global variables

**ERROR_MESSAGE**  The error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is selected.

Usage

**Usage rule**  This component can be used as a standalone component of a Job or Subjob.

Related scenario

No scenario is available for this component yet.
tSSH

Establishes a communication with distant server and returns securely sensible information.
Returns data from a remote computer, based on the secure shell command defined.

**tSSH Standard properties**

These properties are used to configure tSSH running in the Standard Job framework.
The Standard tSSH component belongs to the System family.
The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
• View schema: choose this option to view the schema only.  
• Change to built-in property: choose this option to change the schema to Built-in for local changes.  
• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.  
Click Sync columns to retrieve the schema from the preceding component in the Job. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in</td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Repository</td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and job flowcharts. Related topic: see Talend Studio User Guide</td>
</tr>
<tr>
<td>Host</td>
<td>IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number</td>
</tr>
<tr>
<td>User</td>
<td>User authentication information</td>
</tr>
<tr>
<td>Public Key/Key Passphrase/Private Key</td>
<td>Select the relevant option.</td>
</tr>
</tbody>
</table>
In case of **Public Key**, type in the passphrase, if required, in the **Key Passphrase** field and then in the **Private key** field, type in the private key or click the three dot button next to the **Private key** field to browse to it.

To enter the passphrase, click the [...] button next to the passphrase field, and then in the pop-up dialog box enter the passphrase between double quotes and click **OK** to save the settings.

<table>
<thead>
<tr>
<th><strong>Password/Password</strong></th>
<th>Select the relevant option.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>In case of</strong> <strong>Password</strong>, type in the required password in the <strong>Password</strong> field.</td>
<td></td>
</tr>
<tr>
<td><strong>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Keyboard Interactive/Password</strong></th>
<th>Select the relevant option.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>In case of</strong> <strong>Keyboard Interactive</strong>, type in the required password in the <strong>Password</strong> field.</td>
<td></td>
</tr>
<tr>
<td><strong>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Pseudo terminal</strong></th>
<th>Select this check box to call the interactive shell that performs the terminal operations.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command separator</strong></td>
<td>Type in the command separator required. Once the <strong>Pseudo terminal</strong> check box is selected, this field becomes unavailable.</td>
</tr>
<tr>
<td><strong>Commands</strong></td>
<td>Type in the command for the relevant information to be returned from the remote computer. When you select the <strong>Pseudo terminal</strong> check box, this table becomes a terminal emulator and each row in this table is a single command.</td>
</tr>
<tr>
<td><strong>Use timeout</strong></td>
<td>Define the timeout time period. A timeout message will be generated if the actual response time exceeds this expected processing time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Standard Output</strong></th>
<th>Select the destination to which the standard output is returned. The output may be returned to:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>- to console</strong>: the output is displayed in the console of the <strong>Run view</strong>.</td>
<td></td>
</tr>
<tr>
<td><strong>- to global variable</strong>: the output is indicated by the corresponding global variable.</td>
<td></td>
</tr>
<tr>
<td><strong>- both to console and global variable</strong>: the output is indicated both of the two means.</td>
<td></td>
</tr>
<tr>
<td><strong>- normal</strong>: the output is a standard ssh output.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Error Output</strong></th>
<th>Select the destination to which the error output is returned. The output may be returned to:</th>
</tr>
</thead>
</table>
- **to console**: the output is displayed in the console of the Run view.
- **to global variable**: the output is indicated by the corresponding global variable.
- **both to console and global variable**: the output is indicated both of the two means.
- **normal**: the output is a standard ssh output.

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
**STDOUT**: the standard execution output of the remote command. This is an After variable and it returns a string.  
**STDERR**: the error execution output of the remote command. This is an After variable and it returns a string.  
**EXIT_CODE**: the exit code of the remote command. This is an After variable and it returns an integer.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see **Talend Studio User Guide**. |

**Usage**

| Usage rule | This component can be used as standalone component. Note that for some sensitive information such as password and username, you can define the context variables in the **Contexts** tab and press **Ctrl + Space** to access them. |
| Connections | Outgoing links (from this component to another):  
**Row**: Main  
**Trigger**: Run if; On Component Ok; On Component Error; On Subjob Ok; On Subjob Error. |
**Scenario: Remote system information display via SSH**

The following use case describes a basic Job that uses SSH command to display the hostname of the distant server being connected to, and the current date on this remote system.

The `tSSH` component is sufficient for this Job. Drop it from the Palette to the design workspace.

Double-click on the `tSSH` component and select the Basic settings view tab.

**Procedure**

1. Type in the name of the **Host** to be accessed through SSH as well as the **Port** number.
2. Fill in the **User** identification name on the remote machine.
3. Select the **Authentication method** on the list. For this use case, the authentication method used is the public key.
4. Thus fill in the corresponding **Private key**.

5. On the **Command** field, type in the following command. For this use case, type in `hostname; date` between double quotes.

6. Select the **Use timeout** check box and set the time before falling in error to 5 seconds.

```
picasso
Wed Sep  26 14:24:15 CEST 2007
Job uniteElisa ended at 16:26 26/09/2007. [exit code=0]
```

**Results**

The remote machine returns the host name and the current date and time as defined on its system.
tStatCatcher

Gathers the Job processing metadata at the Job level and at the component level and transfers the log data to the subsequent component for display or storage.

Based on the pre-defined schema, tStatCatcher gathers the Job processing metadata at the Job level and at the component level when the **tStatCatcher** Statistics check box is selected.

**tStatCatcher Standard properties**

These properties are used to configure tStatCatcher running in the Standard Job framework.

The Standard tStatCatcher component belongs to the Logs & Errors family.

The component in this framework is generally available.

**Basic settings**

| Schema   | A schema is a row description, it defines the fields to be processed and passed on to the next component. In this particular case, the schema is read-only, as this component gathers standard log information including:
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| **Moment**: Processing time and date | **Pid**: Process ID
| **Father_pid**: Process ID of the father Job if applicable. If not applicable, Pid is duplicated. | **Root_pid**: Process ID of the root Job if applicable. If not applicable, pid of current Job is duplicated.
| **System_pid**: Thread ID. | **Project**: Project name, which the Job belongs to.
| **Job**: Name of the current Job | **Job_repository_id**: ID of the Job’s .item file stored in the repository.
| **Job_version**: Version of the current Job. | **Context**: Name of the current context
| **Origin**: Name of the component if any | **Message_type**: Begin or End.
| **Message**: Success or Failure. | **Duration**: Time for the execution of a Job or a component with the **tStatCatcher** Statistics check box selected. |
Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see  Talend Studio  User Guide. |

Usage

| Usage rule | This component is the start component of a secondary Job which triggers automatically at the end of the main Job. The processing time is also displayed at the end of the log. |

Scenario: Displaying the statistics log of Job execution

This scenario collects the statistics log for the Job execution and displays it on the Run console. Note that, since the tStatCatcher Statistics check box is not selected for the components, the statistics log applies solely to this specific Job.

Linking the components

Procedure

1. Drop tFixedFlowInput, tFileOutputDelimited, tStatCatcher and tLogRow onto the workspace.
2. Link tFixedFlowInput to tFileOutputDelimited using a Row > Main connection.
3. Link tStatCatcher to tLogRow using a Row > Main connection.
Configuring the components

Procedure

1. Double-click **tFixedFlowInput** to open its **Basic settings** view.

2. Click the **Edit schema** button to open the schema editor.

3. Click the [+ ] button to add three columns, namely **ID_Owners**, **Name_Customer** and **ID_Insurance**, of the Integer and String types respectively.

4. Click **Ok** to validate the setup and close the editor.

5. In the dialog box that appears, click **Yes** to propagate the changes to the subsequent component.

6. Select the **Use Inline Content (delimited file)** option.

7. In the **Content** box, enter `1;Andrew;888`.

8. Double-click **tFileOutputDelimited** to open its **Basic settings** view.
9. In the **File Name** field, enter the full name of the file to save the statistics data.

10. Double-click **tLogRow** to open its **Basic settings** view.

11. Select **Vertical (each row is a key/value list)** for a better display of the results.

**Executing the Job**

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to run the Job.
As shown above, the statistics log of the Job execution is correctly generated.
tSugarCRMInput

Extracts data from a SugarCRM database based on a query.
tSugarCRMInput connects to a Sugar CRM database module via the relevant webservice.

**tSugarCRMInput Standard properties**

These properties are used to configure tSugarCRMInput running in the Standard Job framework.
The Standard tSugarCRMInput component belongs to the Business and the Cloud families.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>SugarCRM Webservice URL</th>
<th>Type in the webservice URL to connect to the SugarCRM DB.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Username and Password</strong></td>
<td>Type in the Webservice user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>Select the relevant module from the list</td>
</tr>
<tr>
<td></td>
<td>Note: To use customized tables, select Use custom module from the list. The Custom module package name and Custom module name fields which appear are automatically filled in with the relevant names.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• Change to built-in property: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
</tbody>
</table>
In this component the schema is related to the Module selected.

**Query condition**

Type in the query to select the data to be extracted. Example: account_name= 'Talend'.

**Advanced settings**

| **tStat Catcher Statistics** | Select this check box to collect log data at the component level. |

**Global Variables**

| **Global Variables** | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see **Talend Studio User Guide**. |

**Usage**

| **Usage rule** | Usually used as a Start component. An output component is required. |

**Scenario: Extracting account data from SugarCRM**

This scenario describes a two-component Job which extracts account information from a SugarCRM database and writes it to an Excel output file.

**Setting up the Job**

**Procedure**

1. Drop a **tSugarCRMInput** and a **tFileOutputExcel** component from the **Palette** onto the workspace.
2. Connect the input component to the output component using a **Row > Main** connection.
Configuring the input component

Procedure

1. Double-click tSugarCRMInput to define the component properties in its Basic settings view.

   ![SugarCRM Input Component](image1)

<table>
<thead>
<tr>
<th>Basic settings</th>
<th>Advanced settings</th>
<th>Dynamic settings</th>
<th>View</th>
<th>Documentation</th>
<th>Validation Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>SugarCRM WebService URL</td>
<td>Username</td>
<td>Password</td>
<td>Module</td>
<td>Schema</td>
<td>Query Condition</td>
</tr>
<tr>
<td>&quot;<a href="http://localhost/sugar/soap.php">http://localhost/sugar/soap.php</a>&quot;</td>
<td>&quot;admin&quot;</td>
<td>&quot;password&quot;</td>
<td>Accounts</td>
<td>Built-In</td>
<td>&quot;billing_address_city='Sunnyvale'&quot;</td>
</tr>
</tbody>
</table>

2. Fill the SugarCRM WebService URL field with the connection information, and the Username and Password fields with the authentication you have.

3. Select the Module from the list of modules offered. In this example, Accounts is selected.

   The Schema is then automatically set according to the module selected. But you can change it and remove the columns that you do not require in the output.

4. In the Query Condition field, type in the query you want to extract from the CRM. In this example: "billing_address_city='Sunnyvale'".

Job execution

Procedure

1. Double-click tFileOutputExcel to define the component properties in its Basic settings view.

   ![File Output Excel Component](image2)

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>&quot;C:/Output/billing_city.xls&quot;</td>
</tr>
<tr>
<td>Sheet name</td>
<td>&quot;accounts&quot;</td>
</tr>
<tr>
<td>Include header</td>
<td>checked</td>
</tr>
<tr>
<td>Schema Type</td>
<td>Built-In</td>
</tr>
<tr>
<td>Encoding Type</td>
<td>ISO-8859-15</td>
</tr>
</tbody>
</table>

2. Set the destination file name as well as the Sheet name and select the Include header check box.

3. Press CTRL+S to save your Job and press F6 to execute it.
The filtered data is output in the defined spreadsheet of the specified Excel file.
**tSugarCRMOutput**

Writes data into a SugarCRM database.
tSugarCRMOutput writes in a Sugar CRM database module via the relevant webservice.

**tSugarCRMOutput Standard properties**

These properties are used to configure tSugarCRMOutput running in the Standard Job framework. The Standard tSugarCRMOutput component belongs to the Business and the Cloud families. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SugarCRM WebService URL</td>
<td>Type in the webservice URL to connect to the SugarCRM DB.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Type in the Webservice user authentication data. To enter the password,</td>
</tr>
<tr>
<td></td>
<td>click the [...] button next to the password field, and then in the pop-up</td>
</tr>
<tr>
<td></td>
<td>dialog box enter the password between double quotes and click OK to save the</td>
</tr>
<tr>
<td></td>
<td>settings.</td>
</tr>
<tr>
<td>Module</td>
<td>Select the relevant module from the list.</td>
</tr>
<tr>
<td>Action</td>
<td>Insert or Update the data in the SugarCRM module.</td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description. It defines the number of fields (columns)</td>
</tr>
<tr>
<td></td>
<td>to be processed and passed on to the next component. When you create a Spark</td>
</tr>
<tr>
<td></td>
<td>Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>Click Edit schema to make changes to the schema. If the current schema is</td>
</tr>
<tr>
<td></td>
<td>of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• View schema: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• Change to built-in property: choose this option to change the schema to</td>
</tr>
<tr>
<td></td>
<td>Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• Update repository connection: choose this option to change the schema</td>
</tr>
<tr>
<td></td>
<td>stored in the repository and decide whether to propagate the changes to all</td>
</tr>
<tr>
<td></td>
<td>the Jobs upon completion. If you just want to propagate the changes to the</td>
</tr>
<tr>
<td></td>
<td>current Job, you can select No upon completion and choose this schema</td>
</tr>
<tr>
<td></td>
<td>metadata again in the [Repository Content] window.</td>
</tr>
</tbody>
</table>
Click **Sync columns** to retrieve the schema from the previous component connected in the Job.

### Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

### Global Variables

| Global Variables | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |

### Usage

| Usage rule | Used as an output component. An Input component is required. |

### Related scenarios

No scenario is available for the Standard version of this component yet.
tSVNLogInput

Retrieves the information of a specified revision or range of revisions from an SVN repository.

**tSVNLogInput Standard properties**

These properties are used to configure tSVNLogInput running in the Standard Job framework.

The Standard tSVNLogInput component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>URL</strong></td>
<td>Type in the URL to the SVN repository to be accessed.</td>
</tr>
<tr>
<td><strong>Need authentication?</strong></td>
<td>Select this check box if authentication is needed to access the SVN repository.</td>
</tr>
</tbody>
</table>
| **Username and Password** | Enter the username and password to access the SVN repository if authentication is needed.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.  
These two fields appear only when the Need authentication? check box is selected. |
| **From revision**         | Specify a start revision number of the revision range.                     |
| **Only this revision**    | Select this check box if you need to retrieve the information of only the revision specified in the From revision field. |
| **To revision**           | Specify an end revision number of the revision range. Enter 'HEAD' for the latest revision.  
This field disappears when the Only this revision check box is selected. |
| **Schema and Edit schema**| A schema is a row description, it defines the fields to be processed and passed on to the next component.  
The schema of this component is read-only. It describes the properties of the revision information. You can click the [...] button next to Edit schema to view the predefined schema which contains the following fields:  
• revision: the revision number of the log message.  
• author: the author who commits the log message.  
• date: the date when the log message was committed.  
• message: the content of the log message.  
• nb_file_added: the number of added files.  
• nb_file_modified: the number of modified files. |
Advanced settings

### tStatCatcher Statistics

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

Global Variables

| Global Variables | NB_LINE: the number of rows processed. This is an After variable and it returns an integer.  
ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component is usually used as a start component in a Job and it needs an output link. |

Connections

| Outgoing links (from this component to another):  
Row: Main, Iterate  
Trigger: On Subjob Ok, On Subjob Error, Run if, On Component Ok, On Component Error |

| Incoming links (from one component to this one):  
Row: Main, Iterate  
Trigger: On Subjob Ok, On Subjob Error, Run if, On Component Ok, On Component Error |

For further information about connections, see Talend Studio User Guide.

---

### Scenario: Retrieving a log message from an SVN repository

This scenario describes a two-component Job that retrieves the information of a specified revision from an SVN repository, and displays the information on the console.
Procedure

1. Create a new Job and add the following two components by typing their names in the design workspace or dropping them from the Palette: a tSVNLogInput component and a tLogRow component.

2. Connect tSVNLogInput to tLogRow using a Row > Main link.

3. Double-click tSVNLogInput to open its Basic settings view.

   ![tSVNLogInput settings](image)

   - URL: "https://talend.com/svndoc/"
   - Need authentication?: select this check box
   - Username: "lena"
   - Password: **********
   - From revision: 17090
   - Only this revision: select this check box

4. In the URL field, enter the URL to the SVN repository to be accessed. In this example, it is https://talend.com/svndoc/. Note that the login credentials provided here is for demonstration only.

5. Select the Need authentication? check box and fill in the Username and Password fields with your authentication information.

6. Specify a revision or revision range. In this example, we want to retrieve the information of revision 17090 only, so enter 17090 in the From revision field and select the Only this revision check box.

7. Double-click tLogRow to open its Basic settings view.

   ![tLogRow settings](image)

   - Schema: Built-In
   - Field Separator: "\n"
   - Mode: Basic
   - Print header: select this box
   - Print component unique name in front of each output row: select this box
   - Print schema column name in front of each value: select this box
   - Use fixed length for values: select this box

8. Define the way you want the output data to be displayed on the console. In this example, enter "\n" in the Field Separator field to display each field of the revision information in a separate line.
9. Select the **Print schema column name in front of each value** check box to show the name of each field of the revision information at the beginning of each line.

10. Press **Ctrl + S** to save the Job and then **F6** to run the Job.

As shown above, the information of the specified revision is displayed on the console.
tSybaseBulkExec

Gains in performance during Insert operations to a Sybase database.

The tSybaseOutputBulk and tSybaseBulkExec components are generally used together as parts of a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tSybaseOutputBulkExec component, detailed in a separate section. The advantage of using two separate components is that the data can be transformed before it is loaded in the database.

tSybaseBulkExec Standard properties

These properties are used to configure tSybaseBulkExec running in the Standard Job framework.

The Standard tSybaseBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Version</td>
<td>Select the version of the Sybase database to be used from the drop-down list.</td>
</tr>
<tr>
<td>Server</td>
<td>Database server IP address</td>
</tr>
<tr>
<td></td>
<td>Available when you select Bulk Update from the Action on data list in the Advanced settings view of tSybaseBulkExec.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td></td>
<td>Available when you select Bulk Update from the Action on data list in the Advanced settings view of tSybaseBulkExec.</td>
</tr>
<tr>
<td>Database</td>
<td>Database name</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Sybase Schema</td>
<td>Name of the database schema you want to use.</td>
</tr>
<tr>
<td>Bcp Utility</td>
<td>Name of the utility to be used to copy data over to the Sybase server.</td>
</tr>
<tr>
<td>Server</td>
<td>IP address of the database server for the Bcp utility connection.</td>
</tr>
<tr>
<td>Batch size</td>
<td>Number of lines in each processed batch.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time and that the table must exist for the insert operation to succeed.</td>
</tr>
<tr>
<td>Action on table</td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td><strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Create table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td><strong>Create table if not exists</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop table if exists and create</strong>: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td><strong>Truncate table</strong>: The table content is deleted. You do not have the possibility to rollback the operation.</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
</tbody>
</table>

### File Name
Name of the file to be loaded.

**Warning:**
This file should be located on the same machine as the database server.

### Schema and Edit Schema
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

**Built-In:** You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](https://help.talend.com).

**Repository:** You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](https://help.talend.com).

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-In** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Advanced settings

| **Use an interface file** | Select this check box to specify an interface file in the field **Interface file**. |
|**Additional JDBC parameters** | Specify additional connection properties in the existing DB connection, to allow specific character set support. E.G.: CHARSET=KANJISJIS_OS to get support of Japanese characters. |
| Action on data | On the data of the table defined, you can perform:  
**Bulk Insert**: Add multiple entries to the table. If duplicates are found, Job stops.  
**Bulk Update**: Make simultaneous changes to multiple entries. |
| Field Terminator | Character, string or regular expression to separate fields.  
**Warning**:  
*With the row/field separators compliant with the Sybase syntax, this component allows for the use of Sybase-orientated characters, such as \x09.* |
| Row Terminator | String (ex: "\n" in Unix) to separate lines. |
| Head row | Number of head lines to be ignored in the beginning of a file. |
| Encoding | Select the encoding from the list, or enter the encoding between double quotes if it does not exist in the list. This field is compulsory for database data handling. |
| Output | Select the type of output for the standard output of the Sybase database:  
to console,  
to global variable. |
| tStataCatcher statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

### Global Variables

| Global Variables |  
**NB_LINE_DATA**: the number of rows read. This is an After variable and it returns an integer.  
**NB_LINE_BAD**: the number of rows rejected. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  
For further information about variables, see Talend Studio User Guide. |
Usage

Usage rule
This component is mainly used when no particular transformation is required on the data to be loaded onto the database.

Dynamic settings
Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Limitation
The database server/client must be installed on the same machine where the Studio is installed or where the Job using tSybaseBulkExec is deployed, so that the component functions properly.

As opposed to the Oracle dedicated bulk component, no action on data is possible using this Sybase dedicated component.

This component requires installation of its related jar files.

Related scenarios

For tSybaseBulkExec related topics, see:

- Scenario: Inserting transformed data in MySQL database on page 2537.
- Scenario: Truncating and inserting file data into an Oracle database on page 2752.
tSybaseClose

Closes a transaction committed in the connected database.

tSybaseClose Standard properties

These properties are used to configure tSybaseClose running in the Standard Job framework.

The Standard tSybaseClose component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

| Database | Select a type of database from the list and click Apply. |
|Component list | Select the tSybaseConnection component in the list if more than one connection are planned for the current Job. |

Advanced settings

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

Usage

| Usage rule | This component is to be used along with Sybase components, especially with tSybaseConnection and tSybaseCommit. |
|Dynamic settings | Click the [+ ] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-
Related scenario

This component is to be used with `tSybaseConnection` and `tSybaseRollback` components. It is generally used with a `tSybaseConnection` to close a connection for the ongoing transaction.

For a related scenario `tSybaseClose`, see `tMysqlConnection` on page 2475.
**tSybaseCommit**

Commits in one go, using a unique connection, a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

tSybaseCommit validates the data processed through the Job into the connected DB.

**tSybaseCommit Standard properties**

These properties are used to configure tSybaseCommit running in the Standard Job framework.

The Standard tSybaseCommit component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tSybaseConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning:</strong> If you want to use a Row &gt; Main connection to link tSybaseCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStat Catcher Statistics  | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule                | This component is more commonly used with other tSybase components, especially with the tSybaseConnection and tSybaseRollback components. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |

## Related scenario

For **tSybaseCommit** related scenario, see **Scenario: Inserting data in mother/daughter tables** on page 2476.
tSybaseConnection

Opens a connection to the database for a current transaction.

**tSybaseConnection Standard properties**

These properties are used to configure tSybaseConnection running in the Standard Job framework.

The Standard tSybaseConnection component belongs to the Databases and the ELT families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td><strong>DB Version</strong></td>
<td>Select the version of the database.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>The IP address or hostname of the database.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>The listening port number of the database.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>The name of the database.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>The database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>The schema of the database.</td>
</tr>
<tr>
<td><strong>Additional JDBC Parameters</strong></td>
<td>Specify additional JDBC parameters for the database connection being created.</td>
</tr>
</tbody>
</table>
### Use or register a shared DB Connection

Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the **Shared DB Connection Name** field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the **tRunJob** component. Using a shared connection together with a **tRunJob** component with either of these two options enabled will cause your Job to fail.

### Advanced settings

#### Auto Commit

Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

#### tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Usage

#### Usage rule

This component is more commonly used with other tSybase* components, especially with the **tSybaseCommit** and **tSybaseRollback** components.

### Related scenarios

For a **tSybaseConnection** related scenario, see [Scenario: Inserting data in mother/daughter tables](#) on page 2476.
tSybaseInput

Executes a DB query with a strictly defined order which must correspond to the schema definition. tSybaseInput reads a database and extracts fields based on a query. It passes on the field list to the next component via a Main row link.

tSybaseInput Standard properties

These properties are used to configure tSybaseInput running in the Standard Job framework. The Standard tSybaseInput component belongs to the Databases family. The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection. For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view.</td>
<td></td>
</tr>
<tr>
<td>For more information about setting up and storing database connection parameters, see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>

Use an existing connection

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

Note:

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component.
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>DB Version</th>
<th>Select the version of the Sybase database to be used from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Database server IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Sybase Schema</td>
<td>Exact name of the Sybase schema.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td>Built-In:</td>
<td>You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td>Repository:</td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Name of the table to read.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query type and Query</td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Trim all the String/Char columns</th>
<th>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trim column</td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

| Global Variables | **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.  
**QUERY**: the query statement being processed. This is a Flow variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see **Talend Studio User Guide**. |

**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible SQL queries for Sybase databases.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the <strong>Code</strong> field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.</td>
</tr>
</tbody>
</table>
The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

| Limitation | This component requires installation of its related jar files. |

**Related scenarios**

For related topics, see:

- **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489.
tSybaseIQBulkExec

Loads data into a Sybase database table from a flat file or other database table.

**tSybaseIQBulkExec Standard properties**

These properties are used to configure tSybaseIQBulkExec running in the Standard Job framework.

The Standard tSybaseIQBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up <strong>Repository Content</strong> dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td></td>
<td>When reusing an existing Sybase 16 (SQL Anywhere) connection created by a tSybaseConnection component and loading data from a file located on the client side, you need to select Sybase 16 (SQL Anywhere) from the <strong>DB Version</strong> drop-down list to display the <strong>Is client file</strong> check box before selecting this check box.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component.
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

This property is not available when **Sybase IQ 15** is selected from the **DB Version** drop-down list.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB Version</strong></td>
<td>Select the version of the Sybase database to be used from the drop-down list.</td>
</tr>
<tr>
<td></td>
<td>- For Sybase IQ 12 and Sybase IQ 16, the file to be bulk-loaded must be located on the same machine as the Sybase IQ server.</td>
</tr>
<tr>
<td></td>
<td>- For Sybase IQ 15, the file to be bulk-loaded can be located on the client side, however, this means certain setup on the Sybase IQ server. For more information, see <em>Sybase IQ client-side load support enhancements</em>.</td>
</tr>
<tr>
<td></td>
<td>- For Sybase 16 (SQL Anywhere), the file to be bulk-loaded can be located on both the server side and the client side.</td>
</tr>
<tr>
<td></td>
<td>Before loading data into Sybase 16 (SQL Anywhere), SQL Anywhere 17 must be installed. You can download it <a href="#">here</a>. Once the installation is done, you need to check the system variable <em>Path</em> to make sure the installation path of SQL Anywhere 17 has been added to it, and then restart the Studio (in certain case restart the computer) to have the new <em>Path</em> variable take effect.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>The IP address or hostname of the database.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when <strong>Sybase IQ 15</strong> is selected from the <strong>DB Version</strong> drop-down list.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>The listening port number of the database.</td>
</tr>
<tr>
<td></td>
<td>This property is not available when <strong>Sybase IQ 15</strong> is selected from the <strong>DB Version</strong> drop-down list.</td>
</tr>
<tr>
<td><strong>Data Source</strong></td>
<td>Select the type of the data source to be used and complete the corresponding DSN information in the field alongside. The available types are DSN and FILEDSN.</td>
</tr>
<tr>
<td></td>
<td>When FILEDSN is selected, a [...] button is available next to the <strong>Data Source</strong> field to allow you to browse to the data source file of interest.</td>
</tr>
<tr>
<td></td>
<td>This property is available when <strong>Sybase IQ 15</strong> is selected from the <strong>DB Version</strong> drop-down list.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>The name of the database.</td>
</tr>
<tr>
<td><strong>Sybase Schema</strong></td>
<td>The schema of the database.</td>
</tr>
</tbody>
</table>
| **Username and Password** | The database user authentication data.
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| **Table** | The name of the table to be written. |
| **Action on table** | On the table defined, you can perform one of the following operations:
- **Default**: No operation is carried out.
- **Drop and create table**: The table is removed and created again.
- **Create table**: The table does not exist and gets created.
- **Create table if does not exist**: The table is created if it does not exist.
- **Drop table if exists and create**: The table is removed if it already exists and created again.
- **Clear table**: The table content is deleted.
- **Truncate table**: The table content is deleted. You do not have the possibility to rollback the operation. |
| **Local filename** | The path to the file to be loaded. |
| **Is client file** | Select this check box if you need to load data from the client side.
This property is available when **Sybase 16 (SQL Anywhere)** is selected from the **DB Version** drop-down list. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the **Next Component**. When you create a Spark Job, avoid the reserved word `line` when naming the fields.
- **Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and repository. |
decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th><strong>Use Insert-Location</strong></th>
<th>Select this check box and specify the source database table from which data will be imported into the Sybase table.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server</strong></td>
<td>The name of the server that stores data to be imported. This property is available only when the <strong>Use Insert-Location</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>The name of the database that stores data to be imported. This property is available only when the <strong>Use Insert-Location</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>The name of the source schema. This property is available only when the <strong>Use Insert-Location</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>The name of the source table. This property is available only when the <strong>Use Insert-Location</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Use Source Filter</strong></td>
<td>Select this check box and specify the row filter on the source table. This property is available only when the <strong>Use Insert-Location</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Where Clause</strong></td>
<td>The where clause to filter the rows you want to import into the Sybase table. This property is available only when the <strong>Use Source Filter</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Columns Mapping</strong></td>
<td>Complete this table to specify the mapping relationship between the source table columns and the Sybase table columns.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Column</strong>: the name of the Sybase table column. By default, the fields in the <strong>Column</strong> column are same as what they are in the schema.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Columns in Source Table</strong>: the name of the corresponding column in the table from which the data will be imported. This property is available only when the <strong>Use Insert-Location</strong> check box is selected.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **Additional JDBC Parameters** | Specify additional JDBC parameters for the database connection being created. |
| **Lines terminated by** | The character or sequence of characters used to separate lines.  
This property is not available when the **Use Insert-Location** check box is selected. |
|-------------------------|--------------------------------------------------------------------------------------------------|
| **Fields terminated by** | The character, string or regular expression used to separate fields.  
With the row/field separators compliant with the Sybase syntax, this component allows the use of Sybase-oriented separators, such as \x09.  
This property is not available when the **Use Insert-Location** or **Use fixed length** check box is selected. |
| **Use enclosed quotes** | Select this check box to use data enclosure characters.  
This property is not available when the **Use Insert-Location** or **Use fixed length** check box is selected. |
| **Use fixed length** | Select this check box to set a fixed width for data lines.  
This property is not available when the **Use Insert-Location** check box is selected. |
| **Include Header** | Select this check box if the file that will be loaded includes header.  
This property is not available when the **Use Insert-Location** check box is selected. |
| **Blank as null** | Select this check box to store the input zero-length varchars as NULLs instead of blanks in the database during the load operation.  
This property is not available when the **Use Insert-Location** or **Use fixed length** check box is selected. |
| **On file error** | Select the action Sybase performs when an error occurs:  
- **Rollback (default)**: rolls back the entire transaction.  
- **Finish**: finishes the insertions already completed and ends the load operation.  
- **Continue**: skips the error and continues the load operation.  
This property is available when **Sybase 16 (SQL Anywhere)** is selected from the **DB Version** drop-down list. |
| **Message log file** | The path to the MESSAGE LOG file that saves the log information about integrity constraint violations, the types of violations, etc. It must be specified together with the **Row log file** property.  
This property is available when **Sybase 16 (SQL Anywhere)** is selected from the **DB Version** drop-down list. |
<p>| <strong>Row log file</strong> | The path to the ROW LOG file that saves the log information about reject data, the timestamps of |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Message log file</strong></td>
<td>The <strong>Message log file</strong> property must be specified together with the <strong>Message log file</strong> property. It must be specified together with the <strong>Message log file</strong> property. This property is available when <strong>Sybase 16</strong> (SQL Anywhere) is selected from the <strong>DB Version</strong> drop-down list.</td>
</tr>
<tr>
<td><strong>Location Options</strong></td>
<td>Select this check box to display the location options. For more information about the location options, see <a href="http://infocenter.sybase.com/help/index.jsp?topic=%5Ccom.sybase.infocenter.dc00801.1510/html/iqrefso/X315746.htm">http://infocenter.sybase.com/help/index.jsp?topic=\com.sybase.infocenter.dc00801.1510/html/iqrefso/X315746.htm</a>. This property is available only when the <strong>Use Insert-Location</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Encrypted Password</strong></td>
<td>Select this check box to specify the use of Open Client Library default password encryption when connecting to the remote source server. This property is available only when the <strong>Location Options</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Use TDS Packet Size</strong></td>
<td>Select this check box and in the <strong>Packet Size</strong> field displayed specify the TDS packet size in bytes. This property is available only when the <strong>Location Options</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Quoted Identifier</strong></td>
<td>Select this check box and from the <strong>Status</strong> drop-down list displayed specify the setting of the QUOTED_IDENTIFIER option on the remote server. This property is available only when the <strong>Location Options</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Set Isolation Level</strong></td>
<td>Select this check box and from the <strong>Isolation Level</strong> drop-down list displayed select the isolation level for the connection to the remote server. This property is available only when the <strong>Location Options</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Insert Load Options</strong></td>
<td>Select this check box to display the insert load options. For more information about the insert load options, see <a href="http://infocenter.sybase.com/help/index.jsp?topic=%5Ccom.sybase.infocenter.dc00801.1510/html/iqrefso/X315746.htm">http://infocenter.sybase.com/help/index.jsp?topic=\com.sybase.infocenter.dc00801.1510/html/iqrefso/X315746.htm</a>. This property is available only when the <strong>Use Insert-Location</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Limit Number</strong></td>
<td>Specify the maximum number of rows to be inserted into the Sybase IQ table. This property is available only when the <strong>Insert Load Options</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>Notify Number</strong></td>
<td>Specify that you will be notified with a message each time the number of rows are successfully inserted into the Sybase IQ table.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Skip Number</td>
<td>Specify the number of rows to be skipped at the beginning of the source table. This property is available only when the <strong>Insert Load Options</strong> check box is selected.</td>
</tr>
<tr>
<td>Start Row ID</td>
<td>Specify the ID of the row in the Sybase IQ table where you want to start the insert. <strong>Note:</strong> Sybase IQ 16 doesn’t support this parameter. This property is available only when the <strong>Insert Load Options</strong> check box is selected.</td>
</tr>
<tr>
<td>Word Skip Number</td>
<td>Enter the number of times the error <em>Words exceeding the maximum permitted word length not supported</em> is ignored when loading data into the Sybase IQ table. This property is available only when the <strong>Insert Select Load Options</strong> check box is selected.</td>
</tr>
</tbody>
</table>
| Ignore Constraint      | Select this check box and in the table displayed you can specify the integrity constraint violation to ignore when loading data into the Sybase IQ table by clicking the [+] button to add as many rows as needed, each row for a constraint violation, and setting the value of the following columns for each constraint violation:  
  - **Constraint Type:** Click the cell and from the drop-down list displayed select the type of the constraint violation to ignore.  
  - **Number:** Enter the maximum number of the constraint violation to ignore before initiating a rollback during a load.  
This property is available only when the **Insert Select Load Options** check box is selected. |
| Log In File            | Select this check box if you want to log the constraint violation information in files. This property is available only when the **Insert Select Load Options** check box is selected.                          |
| Message Log            | Specify the path to the MESSAGE LOG file. This property is available only when the **Log In File** check box is selected.                                                                                      |
| Row Log                | Specify the path to the ROW LOG file.                                                                                                                                                                        |
This property is available only when the Log In File check box is selected.

**Log Delimited By**
Specify the separator between data values in the ROW LOG file.
This property is available only when the Log In File check box is selected.

**Log Filter**
Select this check box and from the Only Log drop-down list displayed select the type of the constraint violation information to be logged.
This property is available only when the Log In File check box is selected.

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Usage

**Dynamic settings**
Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Related scenarios

For tSybaseIQBulkExec related topics, see:

- **Scenario: Bulk-loading data to a Sybase IQ 12 database** on page 3820.
- **Scenario: Inserting transformed data in MySQL database** on page 2537.
- **Scenario: Truncating and inserting file data into an Oracle database** on page 2752.
tSybaseIQOutputBulkExec

Gains in performance during Insert operations to a Sybase IQ database.
tSybaseIQOutputBulkExec executes the Insert action on the data provided.

**tSybaseIQOutputBulkExec Standard properties**

These properties are used to configure tSybaseIQOutputBulkExec running in the Standard Job framework.

The Standard tSybaseIQOutputBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When reusing an existing Sybase 16 (SQL Anywhere) connection created by a tSybaseConnection component and loading data from a file located on the client side, you need to select Sybase 16 (SQL Anywhere) from the DB Version drop-down list to display the Is client file check box before selecting this check box.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

This property is not available when **Sybase IQ 15** is selected from the **DB Version** drop-down list.

| **DB Version** | Select the version of the Sybase database to be used from the drop-down list.  
|                | • For Sybase IQ 12 and Sybase IQ 16, the file to be bulk-loaded must be located on the same machine as the Sybase IQ server. 
|                | • For Sybase IQ 15, the file to be bulk-loaded can be located on the client side, however, this means certain setup on the Sybase IQ server. For more information, see [Sybase IQ client-side load support enhancements](#).  
|                | • For Sybase 16 (SQL Anywhere), the file to be bulk-loaded can be located on both the server side and the client side.  
|                | Before loading data into Sybase 16 (SQL Anywhere), SQL Anywhere 17 must be installed. You can download it [here](#). Once the installation is done, you need to check the system variable `Path` to make sure the installation path of SQL Anywhere 17 has been added to it, and then restart the Studio (in certain case restart the computer) to have the new `Path` variable take effect. |
| **Host**       | The IP address or hostname of the database.  
|                | This property is not available when **Sybase IQ 15** is selected from the **DB Version** drop-down list. |
| **Port**       | The listening port number of the database.  
|                | This property is not available when **Sybase IQ 15** is selected from the **DB Version** drop-down list. |
| **Data Source**| Select the type of the data source to be used and complete the corresponding DSN information in the field alongside. The available types are **DSN** and **FILEDSN**.  
|                | When **FILEDSN** is selected, a [...] button is available next to the **Data Source** field to allow you to browse to the data source file of interest.  
|                | This property is available when **Sybase IQ 15** is selected from the **DB Version** drop-down list. |
| **Database**   | The name of the database. |
| **Sybase Schema** | The schema of the database. |
| **Username and Password** | The database user authentication data.  
<p>|                | To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings. |</p>
<table>
<thead>
<tr>
<th><strong>Table</strong></th>
<th>The name of the table to be written.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action on table</strong></td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Default</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Drop and create table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Create table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Create table if does not exist</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Drop table if exists and create</strong>: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Clear table</strong>: The table content is deleted.</td>
</tr>
<tr>
<td><strong>Filename</strong></td>
<td>Enter the name of the file to be generated and loaded.</td>
</tr>
<tr>
<td><strong>Is client file</strong></td>
<td>Select this check box if you need to load data from the client side.</td>
</tr>
<tr>
<td></td>
<td>This property is available when Sybase 16 (SQL Anywhere) is selected from the <strong>DB Version</strong> drop-down list.</td>
</tr>
<tr>
<td><strong>Append the file</strong></td>
<td>Select this check box to add the new rows at the end of the records.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td></td>
<td>When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.</td>
</tr>
<tr>
<td></td>
<td>You can find more details about how to verify default values in retrieved schema in <strong>Talend Help Center</strong> (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
</tbody>
</table>
• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

### Advanced settings

<table>
<thead>
<tr>
<th>Additional JDBC Parameters</th>
<th>Specify additional connection properties in the existing DB connection, to allow specific character set support.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields terminated by</td>
<td>Enter the character, string or regular expression used to separate fields.</td>
</tr>
<tr>
<td><strong>Warning:</strong></td>
<td>As a combination of <strong>tSybaseOutputBulk</strong> and <strong>tSybaseIQBulkExec</strong>, this component does not allow the use of Sybase-oriented row/field separators, such as \x09. To achieve the desired effect (for example, displaying fields in the tabular form), you need to use <strong>tSybaseOutputBulk</strong> and <strong>tSybaseIQBulkExec</strong> together to replace <strong>tSybaseIQOutputBulkExec</strong>, with \t used in the former component and \x09 used in the latter.</td>
</tr>
<tr>
<td>Lines terminated by</td>
<td>Enter the character or sequence of characters used to separate lines.</td>
</tr>
<tr>
<td>Use enclosed quotes</td>
<td>Select this check box to use data enclosure characters.</td>
</tr>
<tr>
<td>Include Head</td>
<td>Select this check box to include the column header.</td>
</tr>
<tr>
<td>Blank as null</td>
<td>Select this check box to store the input zero-length varchars as NULLs instead of blanks in the database during the load operation.</td>
</tr>
<tr>
<td>On file error</td>
<td>Select the action Sybase performs when an error occurs:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Rollback (default)</strong>: rolls back the entire transaction.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Finish</strong>: finishes the insertions already completed and ends the load operation.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Continue</strong>: skips the error and continues the load operation.</td>
</tr>
<tr>
<td></td>
<td>This property is available when <strong>Sybase 16 (SQL Anywhere)</strong> is selected from the <strong>DB Version</strong> drop-down list.</td>
</tr>
<tr>
<td>Message log file</td>
<td>The path to the MESSAGE LOG file that saves the log information about integrity constraint violations, the types of violations, etc. It must be specified together with the <strong>Row log file</strong> property.</td>
</tr>
</tbody>
</table>
This property is available when Sybase 16 (SQL Anywhere) is selected from the DB Version drop-down list.

**Row log file**
The path to the ROW LOG file that saves the log information about reject data, the timestamps of the start and completion of the load, etc. It must be specified together with the Message log file property.
This property is available when Sybase 16 (SQL Anywhere) is selected from the DB Version drop-down list.

**Encoding**
Select the encoding type from the list or select Custom and define it manually. This field is compulsory for DB data handling.

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Usage**

**Dynamic settings**
Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

**Usage rule**
This component is mainly used when no particular transformation is required on the data to be loaded onto the database.

**Scenario: Bulk-loading data to a Sybase IQ 12 database**

This scenario describes a Job that writes data generated by a tRowGenerator component into a file and then bulk-loads the data from the file to a Sybase IQ 12 database.
Adding and linking the components

Procedure

1. Create a new Job and add a tRowGenerator component and a tSybaseIQOutputBulkExec component by typing their names in the design workspace or dropping them from the Palette.
2. Link the tRowGenerator component to the tSybaseIQOutputBulkExec component using a Row > Main connection.

Configuring the components

Procedure

1. Double-click the tRowGenerator component to open its RowGenerator Editor.

2. Click the [+] button to add two columns and set their properties:
   - id, Integer type, using `Numeric.sequence(String,int,int)` from the Functions list to generate sequence numbers.
   - name, String type, using `TalendDataGenerator.getFirstName()` from the Functions list to generate random first names.

3. In the Number of Rows for RowGenerator field, enter the number of rows to be generated. In this example, keep the default value 100.
4. Click OK to close the editor and in the pop-up dialog box click Yes to propagate the schema to the next component.
5. Double-click the tSybaseIQOutputBulkExec component to open its Basic settings view.

6. In the Host, Port, Database, Username and Password fields, enter the information required to access the Sybase IQ 12 database.

7. In the Table field, enter the name of the table into which the data will be written. In this example, it is staff.

8. In the Action on table list, select Create table if not exists.

9. In the Filename field, enter the full path of the file to hold the data.

**Saving and executing the Job**

**Procedure**

1. Press Ctrl+S to save the Job.

2. Press F6 to run the Job.

   Starting job usecase_sybase at 14:50 14/11/2012.

   [statistics] connecting to socket on port 3643
   [statistics] connected
   [statistics] disconnected
   Job usecase_sybase ended at 14:50 14/11/2012. [exit code=0]

3. In the Sybase Central console, open the table staff to check the data:
As shown above, the table is created with data inserted.

**Related scenarios**

For use cases in relation with `tSybaseIQOutputBulkExec`, see the following scenarios:

- **Scenario:** Inserting transformed data in MySQL database on page 2537.
- **Scenario:** Inserting data in bulk in MySQL database on page 2544.
- **Scenario:** Truncating and inserting file data into an Oracle database on page 2752.
tSybaseOutput

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the job.
tSybaseOutput writes, updates, makes changes or suppresses entries in a database.

**tSybaseOutput Standard properties**

These properties are used to configure tSybaseOutput running in the Standard Job framework.
The Standard tSybaseOutput component belongs to the Databases family.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**Note:**
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see *Talend Studio User Guide*.

### DB Version
Select the version of the Sybase database to be used from the drop-down list.

### Server
Database server IP address

### Port
Listening port number of DB server.

### Database
Name of the database

### Sybase Schema
Exact name of the Sybase schema.

### Username and Password
DB user authentication data.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

### Table
Name of the table to be written. Note that only one table can be written at a time.

### Action on table
On the table defined, you can perform one of the following operations:

- **Default**: No operation is carried out.
- **Drop and create table**: The table is removed and created again.
- **Create table**: The table does not exist and gets created.
- **Create table if not exists**: The table is created if it does not exist.
- **Drop table if exists and create**: The table is removed if it already exists and created again.
- **Clear table**: The table content is deleted.
- **Truncate table**: The table content is deleted. You do not have the possibility to rollback the operation.

### Turn on identity insert
Select this check box to use your own sequence for the identity value of the inserted records (instead of having the SQL Server pick the next sequential value).

### Action on data
On the data of the table defined, you can perform:

- **Insert**: Add new entries to the table. If duplicates are found, job stops.
- **Update**: Make changes to existing entries.
**Insert or update:** Insert a new record. If the record with the given reference already exists, an update would be made.

**Update or insert:** Update the record with the given reference. If the record does not exist, a new record would be inserted.

**Delete:** Remove entries corresponding to the input flow.

**Warning:**

*It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking Edit Schema and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the Advanced settings view where you can simultaneously define primary keys for the Update and Delete operations. To do that: Select the Use field options check box and then in the Key in update column, select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the Key in delete column for the Delete operation.*

---

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

### Built-In

You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

### Repository

You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

### Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to Built-in for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate...
the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

<table>
<thead>
<tr>
<th><strong>Die on error</strong></th>
<th>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</th>
</tr>
</thead>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Commit every</strong></th>
<th>Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional Columns</strong></td>
<td>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</td>
</tr>
<tr>
<td><strong>Name:</strong></td>
<td>Type in the name of the schema column to be altered or inserted as new column</td>
</tr>
<tr>
<td><strong>SQL expression:</strong></td>
<td>Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td><strong>Position:</strong></td>
<td>Select <strong>Before</strong>, <strong>Replace</strong> or <strong>After</strong> following the action to be performed on the reference column.</td>
</tr>
<tr>
<td><strong>Reference column:</strong></td>
<td>Type in a column of reference that the <strong>tDBOutput</strong> can use to place or replace the new or altered column.</td>
</tr>
<tr>
<td><strong>Use field options</strong></td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td><strong>Enable debug mode</strong></td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td><strong>Use Batch</strong></td>
<td>Select this check box to activate the batch mode for data processing.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>This check box is available only when you have selected the <strong>Insert</strong>, the <strong>Update</strong> or the <strong>Delete</strong> option in the <strong>Action on data</strong> field.</td>
</tr>
<tr>
<td><strong>Batch Size</strong></td>
<td>Specify the number of records to be processed in each batch. This field appears only when the <strong>Use batch mode</strong> check box is selected.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
### Global Variables

<table>
<thead>
<tr>
<th>Global Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NB_LINE</strong></td>
<td>the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_UPDATED</strong></td>
<td>the number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_INSERTED</strong></td>
<td>the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_DELETED</strong></td>
<td>the number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>NB_LINE_REJECTED</strong></td>
<td>the number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td><strong>QUERY</strong></td>
<td>the query statement processed. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

### Usage

#### Usage rule

This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a Sybase database. It also allows you to create a reject flow using a **Row > Rejects** link to filter data in error. For an example of [tMySqlOutput](#) in use, see Scenario: Retrieving data in error with a Reject link on page 2529.

#### Dynamic settings

Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario:
Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

| Limitation          | This component requires installation of its related jar files. |

### Related scenarios

For use cases in relation with **tSybaseOutput**, see:

- **Scenario: Inserting a column and altering data using tMysqlOutput** on page 2521.
tSybaseOutputBulk

Prepares the file to be used as parameter in the INSERT query to feed the Sybase database.

The tSybaseOutputBulk and tSybaseBulkExec components are generally used together as parts of a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tSybaseOutputBulkExec component, detailed in a separate section. The advantage of using two separate components is that the data can be transformed before it is loaded in the database.

 Writes a file with columns based on the defined delimiter and the Sybase standards.

**tSybaseOutputBulk Standard properties**

These properties are used to configure tSybaseOutputBulk running in the Standard Job framework.

The Standard tSybaseOutputBulk component belongs to the Databases family.

The component in this framework is generally available.

*Note:*

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see *Dynamic database components* on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>Judith</td>
<td>- <strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>- <strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>File Name</td>
<td>Name of the file to be generated.</td>
</tr>
<tr>
<td>Warning</td>
<td>This file is generated on the local machine or a shared folder on the LAN.</td>
</tr>
<tr>
<td>Append</td>
<td>Select this check box to add the new rows at the end of the file.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <strong>line</strong> when naming the fields.</td>
</tr>
</tbody>
</table>
### Advanced settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row separator</td>
<td>String (ex: <code>\n</code> on Unix) to distinguish rows.</td>
</tr>
<tr>
<td>Field separator</td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td>Include header</td>
<td>Select this check box to include the column header in the file.</td>
</tr>
<tr>
<td>Encoding</td>
<td>Select the encoding from the list or select Custom and define it manually.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### tSybaseOutputBulk

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see <a href="https://help.talend.com">Talend Studio User Guide</a>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <a href="https://help.talend.com">Talend Studio User Guide</a>. When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually. You can find more details about how to verify default values in retrieved schema in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-In** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.

---

**Warning:**

*Fully in line with the Java syntax, this component does not allow the use of Sybase-orientated row/field separators, such as `\x09`.*
Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component requires installation of its related jar files.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This component is to be used along with tSybaseBulkExec component. Used together they offer gains in performance while feeding a Sybase database.</td>
</tr>
</tbody>
</table>

Component family | Databases/Sybase

Related scenarios

For use cases in relation with tSybaseOutputBulk, see the following scenarios:

- Scenario: Inserting transformed data in MySQL database on page 2537.
- Scenario: Inserting data in bulk in MySQL database on page 2544.
- Scenario: Truncating and inserting file data into an Oracle database on page 2752.
tSybaseOutputBulkExec

Gains in performance during Insert operations to a Sybase database.

The tSybaseOutputBulk and tSybaseBulkExec components are generally used together as parts of a two step process. In the first step, an output file is generated. In the second step, this file is used in the INSERT operation used to feed a database. These two steps are fused together in the tSybaseOutputBulkExec component.

tSybaseOutputBulkExec Standard properties

These properties are used to configure tSybaseOutputBulkExec running in the Standard Job framework.

The Standard tSybaseOutputBulkExec component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

Note:

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>DB Version</th>
<th>Select the version of the Sybase database to be used from the drop-down list.</th>
</tr>
</thead>
</table>
| Server              | Database server IP address
Currenty, only *localhost, 127.0.0.1 or the exact IP address of the local machine is allowed for proper functioning. In other words, the database server must be installed on the same machine where the Studio is installed or where the Job using tSybaseOutputBulkExec is deployed. |
| Port                | Listening port number of DB server. |
| Database            | Name of the database |
| Username and Password| DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Bcp utility         | Name of the utility to be used to copy data over to the Sybase server. |
| Batch row number    | Number of lines in each processed batch. |
| Table               | Name of the table to be written. Note that only one table can be written at a time and that the table must exist for the insert operation to succeed. |
| Action on table     | On the table defined, you can perform one of the following operations:
- **None**: No operation is carried out.
- **Drop and create a table**: The table is removed and created again.
- **Create a table**: The table does not exist and gets created.
- **Create a table if not exists**: The table is created if it does not exist.
- **Clear a table**: The table content is deleted. |
| File Name           | Name of the file to be generated and loaded.

> **Warning:**
This file is generated on the machine specified by the URI in the Server field so it should be on the same machine as the database server.
### Append
Select this check box to add the new rows at the end of the records.

### Schema and Edit Schema
A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

<table>
<thead>
<tr>
<th>Built-In</th>
<th>Description</th>
<th>Related topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>You create and store the schema locally for this component only.</td>
<td>Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Repository</th>
<th>Description</th>
<th>Related topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs.</td>
<td>Related topic: see Talend Studio User Guide.</td>
<td></td>
</tr>
</tbody>
</table>

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

### Advanced settings

#### Use an interface file
Select this check box to specify an interface file in the field Interface file.

#### Additional JDBC parameters
Specify additional connection properties in the existing DB connection, to allow specific character set support. E.G.: CHARSET=KANJISJIS_OS to get support of Japanese characters.

#### Action on data
On the data of the table defined, you can perform:
- **Bulk Insert**: Add multiple entries to the table. If duplicates are found, job stops.
### Bulk Update

- **Field terminator**: Character, string or regular expression to separate fields.
  - **Warning**: As a combination of `tSybaseOutputBulk` and `tSybaseBulkExec`, this component does not allow the use of Sybase-oriented row/field separators, such as `\x09`. To achieve the desired effect (for example, displaying fields in the tabular form), you need to use `tSybaseOutputBulk` and `tSybaseBulkExec` together to replace `tSybaseOutputBulkExec`, with `\t` used in the former component and `\x09` used in the latter.

- **DB Row terminator**: String (ex: `\n` on Unix) to distinguish rows in the DB.

- **First row NO. of file**: Type in the number of the file row where the action should start at.

- **FILE Row terminator**: Character, string or regular expression to separate fields in a file.

- **Include Head**: Select this checkbox to include the column header.

- **Encoding**: Regarding the source data, select the appropriate encoding from the list or select Custom and define it manually. This field is compulsory for DB data handling.

- **Sybase encoding type**: Select the Sybase-specific encoding type for the data to be processed. This encoding type allows `tSybaseOutputBulk` to properly generate the Sybase Bulk command.

- **Output**: Select the type of output for the standard output of the Sybase database:
  - to console,
  - to global variable.

- **tStatCatcher Statistics**: Select this checkbox to collect log data at the component level.

### Usage

- **Usage rule**: This component is mainly used when no particular transformation is required on the data to be loaded onto the database.

- **Dynamic settings**: Click the `[+]` button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example,
when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Limitation

This component requires installation of its related jar files.

The database server/client must be installed on the same machine where the Studio is installed or where the Job using tSybaseOutputBulkExec is deployed, so that the component functions properly.

## Related scenarios

For use cases in relation with tSybaseOutputBulkExec, see the following scenarios:

- Scenario: Inserting transformed data in MySQL database on page 2537.
- Scenario: Inserting data in bulk in MySQL database on page 2544.
- Scenario: Truncating and inserting file data into an Oracle database on page 2752.
tSybaseRollback

 Cancels the transaction committed in the Sybase database.

 **tSybaseRollback Standard properties**

 These properties are used to configure tSybaseRollback running in the Standard Job framework.

 The Standard tSybaseRollback component belongs to the Databases family.

 The component in this framework is generally available.

 ![Note symbol]  

 **Note:**

 This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

 For more information about dynamic database connectors, see [Dynamic database components on page 598](#).

 **Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tSybaseConnection</strong> component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

 **Advanced settings**

| tStat Catcher Statistics | Select this check box to collect log data at the component level. |

 **Usage**

| Usage rule | This component is more commonly used with other tSybase* components, especially with the **tSybaseConnection** and **tSybaseCommit** components. |
| Dynamic settings | Click the [+1] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  

 Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable. |
Related scenarios

For **tSybaseRollback** related scenario, see Scenario: Rollback from inserting data in mother/daughter tables on page 2480.
tSybaseRow

Acts on the actual DB structure or on the data (although without handling data).

tSybaseRow is the specific component for this database query. It executes the SQL query stated onto the specified database. The row suffix means the component implements a flow in the job design although it doesn’t provide output.

The SQLBuilder tool helps you write easily your SQL statements.

**tSybaseRow Standard properties**

These properties are used to configure tSybaseRow running in the Standard Job framework.

The Standard tSybaseRow component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
<table>
<thead>
<tr>
<th><strong>DB Version</strong></th>
<th>Select the version of the Sybase database to be used from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server</strong></td>
<td>Database server IP address</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Sybase Schema</strong></td>
<td>Exact name of the sybase schema.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table Name</strong></td>
<td>Name of the table to be processed.</td>
</tr>
<tr>
<td><strong>Turn on identity insert</strong></td>
<td>Select this check box to use your own sequence for the identity value of the inserted records (instead of having the SQL Server pick the next sequential value).</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Edit schema</strong></td>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to Built-in for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td><strong>Query type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td><strong>Built-in</strong>: Fill in manually the query statement or build it graphically using SQLBuilder</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Query</strong></th>
<th>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Propagate QUERY’s recordset</strong></th>
<th>Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the <strong>use column</strong> list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>i Note:</strong></td>
<td>This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of <strong>Object</strong> and this component is usually followed by <strong>tParseRecordSet</strong>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Use PreparedStatement</strong></th>
<th>Select this check box if you want to query the database using a PreparedStatement. In the <strong>Set PreparedStatement Parameter</strong> table, define the parameters represented by ‘?’ in the SQL instruction of the <strong>Query</strong> field in the <strong>Basic Settings</strong> tab.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter Index</strong>: Enter the parameter position in the SQL instruction.</td>
<td><strong>Parameter Type</strong>: Enter the parameter type.</td>
</tr>
<tr>
<td><strong>Parameter Value</strong>: Enter the parameter value.</td>
<td><strong>Note:</strong> This option is very useful if you need to execute the same query several times. Performance levels are increased</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Commit every</strong></th>
<th>Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
## Global Variables

| Global Variables | **QUERY**: the query statement being processed. This is a Flow variable and it returns a string.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |

## Usage

| Usage rule | This component offers the flexibility of the DB query and covers all possible SQL queries. |
| Dynamic settings | Click the [+•] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.  
The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.  
For examples on using dynamic parameters, see *Scenario: Reading data from databases through context-based dynamic connections* on page 2497 and *Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters* on page 489.  
For more information on **Dynamic settings** and context variables, see Talend Studio User Guide. |

## Limitation

| Limitation | This component requires installation of its related jar files. |

## Related scenarios

For **tSybaseRow** related topics, see:
• **Scenario: Combining two flows for selective output** on page 2558.
• **Procedure** on page 625.
• **Scenario: Removing and regenerating a MySQL table index** on page 2552.
tSybaseSCD

Addresses Slowly Changing Dimension needs, reading regularly a source of data and logging the changes into a dedicated SCD table.

tSybaseSCD reflects and tracks changes in a dedicated Sybase SCD table.

**tSybaseSCD Standard properties**

These properties are used to configure tSybaseSCD running in the Standard Job framework.

The Standard tSybaseSCD component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the Repository file where Properties are stored. The following fields are pre-filled in using fetched data.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.
2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>DB Version</strong></th>
<th>Select the version of the Sybase database to be used from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
</tbody>
</table>
| **Username** and **Password** | DB user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings. |
| **Table**      | Name of the table to be written. Note that only one table can be written at a time. |

**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

**Built-in**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*.

**SCD Editor**

The SCD editor helps to build and configure the data flow for slowly changing dimension outputs.

For more information, see *SCD management methodology* on page 2566.

**Use memory saving Mode**

Select this check box to maximize system performance.
**Die on error**

This check box is cleared by default, meaning to skip the row on error and to complete the process for error-free rows.

**Advanced settings**

**Additional JDBC parameters**

Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the **Basic settings**.

**End date time details**

Specify the time value of the SCD end date time setting in the format of \textit{HH:mm:ss}. The default value for this field is 12:00:00.

This field appears only when SCD Type 2 is used and **Fixed year value** is selected for creating the SCD end date.

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

**Debug mode**

Select this check box to display each step during processing entries in a database.

**Global Variables**

**Global Variables**

**NB_LINE_UPDATED**: the number of rows updated. This is an After variable and it returns an integer.

**NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.

**NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer.

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press \textbf{Ctrl + Space} to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.

**Usage**

**Usage rule**

This component is used as Output component. It requires an Input component and Row main link as input.
| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings and context variables**, see Talend Studio User Guide. |

| Limitation | This component does not support using SCD type 0 together with other SCD types. |

**Related scenarios**

For related topics, see tMysqlSCD on page 2563.
tSybaseSCDELT

Addresses Slowly Changing Dimension needs through SQL queries (server-side processing mode), and logs the changes into a dedicated Sybase SCD table.

tSybaseSCDELT reflects and tracks changes in a dedicated Sybase SCD table.

**tSybaseSCDELT Standard properties**

These properties are used to configure tSybaseSCDELT running in the Standard Job framework.

The Standard tSybaseSCDELT component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally. Enter properties manually.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file where Properties are stored. The fields that come after are pre-filled in using the fetched data.</td>
<td></td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>DB Version</strong></th>
<th>Select the version of the Sybase database to be used from the drop-down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>The IP address of the database server.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>User authentication data for a dedicated database.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Source table</strong></td>
<td>Name of the input Sybase SCD table.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td>Select to perform one of the following operations on the table defined:</td>
</tr>
<tr>
<td></td>
<td><strong>None</strong>: No action carried out on the table.</td>
</tr>
<tr>
<td></td>
<td><strong>Drop and create table</strong>: The table is removed and created again</td>
</tr>
<tr>
<td></td>
<td><strong>Create table</strong>: A new table gets created.</td>
</tr>
<tr>
<td></td>
<td><strong>Create table if not exists</strong>: A table gets created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td><strong>Clear table</strong>: The table content is deleted. You have the possibility to rollback the operation.</td>
</tr>
<tr>
<td></td>
<td><strong>Truncate table</strong>: The table content is deleted. You don't have the possibility to rollback the operation.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: Choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: Choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: Choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong>.</td>
</tr>
</tbody>
</table>
upon completion and choose this schema metadata again in the [Repository Content] window.

| **Built-in:** | The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide*. |
| **Repository:** | The schema already exists and is stored in the Repository, hence can be reused. Related topic: see *Talend Studio User Guide*. |

**Surrogate Key**
Select the surrogate key column from the list.

**Creation**
Select the method to be used for the surrogate key generation.
For more information regarding the creation methods, see *SCD management methodology* on page 2566.

**Source Keys**
Select one or more columns to be used as keys, to ensure the unicity of incoming data.

**Use SCD Type 1 fields**
Use type 1 if tracking changes is not necessary. SCD Type 1 should be used for typos corrections for example. Select the columns of the schema that will be checked for changes.

**Use SCD Type 2 fields**
Use type 2 if changes need to be tracked down. SCD Type 2 should be used to trace updates for example. Select the columns of the schema that will be checked for changes.

**Start date**: Adds a column to your SCD schema to hold the start date value. You can select one of the input schema columns as Start Date in the SCD table.

**End Date**: Adds a column to your SCD schema to hold the end date value for the record. When the record is currently active, the End Date column shows a null value, or you can select Fixed Year value and fill it in with a fictive year to avoid having a null value in the End Date field.

**Log Active Status**: Adds a column to your SCD schema to hold the true or false status value. This column helps to easily spot the active record.

**Log versions**: Adds a column to your SCD schema to hold the version number of the record.

**Advanced settings**

| **Additional JDBC parameters** | Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the *Use an existing connection* check box in the Basic settings. |
| **Debug mode** | Select this check box to display each step during processing entries in a database. |
### tStat Catcher Statistics

Select this check box to collect log data at the component level.

### Global Variables

| Global Variables | Error message: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see *Talend Studio User Guide.* |

### Usage

| Usage rule | This component is used as an output component. It requires an input component and Row main link as input. |

| Dynamic settings | Click the **[+]** button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see *Talend Studio User Guide.* |

| Limitation | This component requires installation of its related jar files. |
Related Scenario

For related topics, see tMysqlSCD on page 2563 and Tracking data changes using Slowly Changing Dimensions (type 0 through type 3) on page 2569.
tSybaseSP

Calls a Sybase database stored procedure.

**tSybaseSP Standard properties**

These properties are used to configure tSybaseSP running in the Standard Job framework.

The Standard tSybaseSP component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see *Dynamic database components* on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

| Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined. |

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*. |
| **DB Version** | Select the version of the Sybase database to be used from the drop-down list. |
| **Host** | Database server IP address |
| **Port** | Listening port number of DB server. |
| **Database** | Name of the database |
| **Username** and **Password** | DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Schema** and **Edit Schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. |
| **Built-In** | You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide. |
| **Repository** | You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. |
| **SP Name** | Type in the exact name of the Stored Procedure |
| **Is Function / Return result in** | Select this check box, if a value is to be returned. Select on the list the schema column, the value to be returned is based on. |
| **Timeout Interval** | Maximum waiting time for the results of the stored procedure. |
| **Parameters** | Click the Plus button and select the various Schema Columns that will be required by the procedures. Note |
that the SP schema can hold more columns than there are parameters used in the procedure.

Select the Type of parameter:

**IN**: Input parameter

**OUT**: Output parameter/return value

**IN OUT**: Input parameters is to be returned as value, likely after modification through the procedure (function).

**RECORDSET**: Input parameters is to be returned as a set of values, rather than single value.

**Note:**
Check Scenario: Inserting data in mother/daughter tables on page 2476 if you want to analyze a set of records from a database table or DB query and return single records.

### Advanced settings

<table>
<thead>
<tr>
<th>Additional JDBC Parameters</th>
<th>Fill in additional connection properties for the DB connection you are creating. This option is available when the check box <strong>Use an existing connection</strong> is not selected in the Basic settings.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Multiple SELECT Procedure</td>
<td>Select this check box to use procedures which contain multiple SELECT statements.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as intermediary component. It can be used as start component but only input parameters are thus allowed.</th>
</tr>
</thead>
</table>
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the **Use an existing connection** check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-
Related scenarios

For related scenarios, see:

- Scenario: Retrieving personal information using a stored procedure on page 2452.
- Scenario: Using tMysqlSP to find a State Label using a stored procedure on page 2583.
- Scenario: Checking number format using a stored procedure on page 2809.
- Executing a stored procedure using tMDMsp on page 2190.

Check tMysqlConnection on page 2475 as well if you want to analyze a set of records from a database table or DB query and return single records.
tSystem

Calls other system processing commands, already up and running in a larger Job.

**tSystem Standard properties**

These properties are used to configure tSystem running in the Standard Job framework.

The Standard tSystem component belongs to the System family.

The component in this framework is generally available.

### Basic settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use home directory</td>
<td>Select this check box to change the name and path of a dedicated directory.</td>
</tr>
<tr>
<td>Use Single Command</td>
<td>When the required command is very simple, to the degree that, for example, only one parameter is used and without space, select this option to activate its Command field. In this field, enter the simple system command. Note that the syntax is not checked.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning:</strong> In Windows, the MS-DOS commands do not allow you to pass directly from the current folder to the folder containing the file to be launched. To launch a file, you must therefore use an initial command to change the current folder, then a second one to launch the file.</td>
</tr>
<tr>
<td>Use Array Command</td>
<td>Select this option to activate its Command field. In this field, enter the system command in array, one parameter per line. For example, enter the following command with consecutive spaces in array for Linux:</td>
</tr>
<tr>
<td></td>
<td>&quot;cp&quot; &quot;\temp\source.txt&quot; &quot;\temp\copy to/&quot;</td>
</tr>
<tr>
<td>Standard Output and Error Output</td>
<td>Select the type of output for the processed data to be transferred to.</td>
</tr>
<tr>
<td></td>
<td><strong>to console:</strong> data is passed on to be viewed in the Run view.</td>
</tr>
<tr>
<td></td>
<td><strong>to global variable:</strong> data is passed on to an output variable linked to the tSystem component.</td>
</tr>
</tbody>
</table>
### to console and to global variable

Data is passed on to the Run view and to an output variable linked to the tSystem component.

### normal

Data is passed on to the component that comes next.

### Schema and Edit Schema

A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

Click **Sync columns** to retrieve the schema from the preceding component in the Job.

### Built-in

You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](#).

### Repository

You have already created the schema and stored it in the Repository. You can reuse it in various projects and job flowcharts. Related topic: see [Talend Studio User Guide](#).

### Environment variables

Click the [+ ] button to add as many global variables as needed.

- **name**: Enter the syntax of the new variable.
- **value**: Enter a value for this variable according to the context.

### Advanced settings

### tStatCatcher Statistics

Select this check box to collect log data at the component level.

### Global Variables

### Global Variables

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

### Usage

#### Usage rule

This component can typically be used for companies which already implemented other applications that they want to integrate into their processing flow through Talend.

#### Connections

**Outgoing links (from this component to another):**
- **Row:** Main.
- **Trigger:** On Subjob Ok; On Subjob Error; Run if.

**Incoming links (from one component to this one):**
- **Row:** Main; Reject; Iterate.
- **Trigger:** On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error; Synchronize; Parallelize.

For further information regarding connections, see Talend Studio User Guide.

### Scenario: Echo 'Hello World!'

This scenario is one single component tSystem to execute a system command and shows the results in the Run view "console".

To replicate this scenario, proceed as follows:
Procedure

1. Drop a tSystem component from the Palette to the design workspace.
2. Double-click tSystem to open its Component view.

3. Select the Use Single Command option to activate its Command field and type in "cmd /c echo Hello World!".
4. In the Standard Output drop-down list, select to both console and global variable.
5. Press F6 to run this Job.

Results

Starting job tSystem_scenario at 17:26 08/10/2009.

[statistics] connecting to socket on port 3961
[statistics] connected
Hello World!
[statistics] disconnected
Job tSystem_scenario ended at 17:26 08/10/2009. [exit code=0]

The Job executes an echo command and shows the output in the Console of the Run view.
tTeradataClose

Closes the transaction committed in the connected DB.

tTeradataClose Standard properties

These properties are used to configure tTeradataClose running in the Standard Job framework.
The Standard tTeradataClose component belongs to the Databases family.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the t TeradataConnection component in the list if more than one connection are planned for the current Job.</td>
</tr>
</tbody>
</table>

Advanced settings

| tStat Catcher Statistics         | Select this check box to collect log data at the component level. |

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is to be used along with Teradata components, especially with tTeradataConnection and tTeradataCommit.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-</td>
</tr>
</tbody>
</table>
Related scenarios

No scenario is available for the Standard version of this component yet.
tTeradataCommit

Commits in one go, using a unique connection, a global transaction instead of doing that on every row or every batch and thus provides gain in performance.

TeradataCommit validates the data processed through the Job into the connected DB.

tTeradataCommit Standard properties

These properties are used to configure tTeradataCommit running in the Standard Job framework.

The Standard tTeradataCommit component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tTeradataConnection component in the list if more than one connection are planned for the current job.</td>
</tr>
</tbody>
</table>
| Close connection | This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task. **Warning:**

If you want to use a Row > Main connection to link tTeradataCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit. |

Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

Usage

| Usage rule | This component is more commonly used with other tTeradata* components, especially with |
the `tTeradataConnection` and `tTeradataRollback` components.

<table>
<thead>
<tr>
<th>Dynamic settings</th>
</tr>
</thead>
</table>
| **Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.**
| **Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.**
| **For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.**
| **For more information on Dynamic settings and context variables, see Talend Studio User Guide.**

**Related scenario**

For `tTeradataCommit` related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476
**tTeradataConnection**

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

tTeradataConnection opens a connection to the database for a current transaction.

**tTeradataConnection Standard properties**

These properties are used to configure tTeradataConnection running in the Standard Job framework.

The Standard tTeradataConnection component belongs to the Databases and the ELT families.

The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see [Dynamic database components](#) on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>DB user authentication data. To enter the password, click the <code>[...]</code> button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Additional JDBC parameters</strong></td>
<td>Specify additional connection properties in the existing DB connection, to allow specific character set support. E.G.: <code>CHARSET=KANJISJIS_OS</code> to get support of Japanese characters. <strong>Note:</strong> You can set the encoding parameters through this field.</td>
</tr>
<tr>
<td><strong>Use or register a shared DB Connection</strong></td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or...</td>
</tr>
</tbody>
</table>
child Job, and in the **Shared DB Connection Name** field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the **tRunJob** component. Using a shared connection together with a **tRunJob** component with either of these two options enabled will cause your Job to fail.

### Advanced settings

#### Query band

Select this check box to use the Teradata Query Banding feature to add metadata to the query to be processed, such as the user running the query. This can help you, for example, identify the origin of this query.

Once selecting the check box, the **Query Band parameters** table is displayed, in which you need to enter the metadata information to be added. This information takes the form of key/value pairs, for example, *DpiID* in the **Key** column and *Finance* in the **Value** column.

This check box actually generates the **SET QUERY_BAND FOR SESSION** statement with the key/value pairs declared in the **Query Band parameters** table. For further information about this statement, see [http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/SQL_Reference/B035_1144_111A/End_Lo](http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/SQL_Reference/B035_1144_111A/End_Lo).

#### Auto Commit

Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

#### tStatCatcher Statistics

Select this check box to collect log data at the component level.
Usage

Usage rule
This component is more commonly used with other tTeradata™ components, especially with the tTeradataCommit and tTeradataRollback components.

Limitation
Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Related scenario
For tTeradataConnection related scenario, see tMysqlConnection on page 2475.
**tTeradataFastExport**

Exports data batches from a Teradata table to a customer system or to a smaller database. tTeradataFastExport exports rapidly voluminous data batches from a Teradata table or view.

**tTeradataFastExport Standard properties**

These properties are used to configure tTeradataFastExport running in the Standard Job framework. The Standard tTeradataFastExport component belongs to the Databases family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Commandline</td>
<td>Select this check box to enable the commandline mode.</td>
</tr>
<tr>
<td>Use Java API</td>
<td>Select this check box to enable the Java API mode.</td>
</tr>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Execution platform</td>
<td>Select the Operating System type you use.</td>
</tr>
<tr>
<td></td>
<td>Available in the <strong>Use Commandline</strong> mode.</td>
</tr>
<tr>
<td>Host</td>
<td>Server name or IP.</td>
</tr>
<tr>
<td>Database name</td>
<td>Database name.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various</td>
</tr>
</tbody>
</table>
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

| **Use query** | Select this check box to show the **Query** box where you can enter the SQL statement. Available in the **Use Commandline** mode. |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| **Query**     | Enter the SQL statement in the **Query** box.                                                                                                                                                     |
| **Log database** | Log database name. Available in the **Use Commandline** mode.                                                                                                                                      |
| **Log table** | Log table name. Available in the **Use Commandline** mode.                                                                                                                                 |
| **Script generated folder** | Browse your directory and select the destination of the file which will be created. Available in the **Use Commandline** mode.                                                                    |
| **Exported file** | Name and path to the file which will be created.                                                                                                                                                  |
| **Field separator** | Character, string or regular expression to separate fields.                                                                                                                                         |
| **Row separator** | String (e.g.: \n on Unix) to separate rows. Available in the **Use Java API** mode.                                                                                                               |
| **Error file** | Browse your directory and select the destination of the file where the error messages will be recorded. Available in the **Use Commandline** mode.                                                   |

### Advanced settings

**Output**

Two options are available in the list:
- **Output error to file**: outputs the error to the file specified in the **Error log** field and continues the processing.
**Output error to console:** outputs the error to the console and ends the processing.
Available in the **Use Commandline** mode.

**tStatCatcher Statistics**
Select this check box to collect log data at the component level.

### Usage

**Usage rule**
This component offers the flexibility benefit of the DB query and covers all possible SQL queries.

**Limitation**
If you have selected the **Use Commandline** mode, you need to install the Teradata client on the machine where there is the Job that involves this component.

### Related scenarios

No scenario is available for the Standard version of this component yet.
tTeradataFastLoad

Executes a database query according to a strict order which must be the same as the one in the schema.

tTeradataFastLoad reads a database and extracts fields using queries. The retrieve list of fields is then transferred to the next component, using a connection flow (Main row).

**tTeradataFastLoad Standard properties**

These properties are used to configure tTeradataFastLoad running in the Standard Job framework. The Standard tTeradataFastLoad component belongs to the Databases family. The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Host</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Database</td>
<td>Database name.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
<tr>
<td>Execute Batch every</td>
<td>Number of rows per batch to be loaded.</td>
</tr>
<tr>
<td>Die on error</td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row &gt; Rejects link.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:</td>
</tr>
</tbody>
</table>
- **View schema**: choose this option to view the schema only.

- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Additional JDBC parameters</strong></th>
<th>Specify additional connection properties for the DB connection you are creating.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*. |
**Usage**

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility benefit of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

**Related scenarios**

No scenario is available for the Standard version of this component yet.
**tTeradataFastLoadUtility**

Executes a database query according to a strict order which must be the same as the one in the schema.

*tTeradataFastLoadUtility* reads a database and extracts fields using queries. The retrieve list of fields is then transferred to the next component, using a connection flow (*Main row*).

**tTeradataFastLoadUtility Standard properties**

These properties are used to configure *tTeradataFastLoadUtility* running in the Standard Job framework.

The Standard *tTeradataFastLoadUtility* component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Execution platform</strong></td>
<td>Select the Operating System type you use.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Host name or IP address of the database server.</td>
</tr>
<tr>
<td><strong>Database name</strong></td>
<td>Database name.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
<tr>
<td><strong>Schema</strong> and <strong>Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various</td>
</tr>
</tbody>
</table>
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Script generated folder**

Browse your directory and select the destination of the file which will be created.

**Load file**

Browse your directory and select the file from which you want to load data.

**Field separator**

Character, string or regular expression to separate fields.

**Error file**

Browse your directory and select the destination of the file where the error messages will be recorded.

### Advanced settings

**Define character set**

Specify the character encoding you need use for your system.

**Check point**

Enter the check point value.

**Error files**

Enter the file name where the error messages are stored. By default, the code `ERRORFILES table_ERR1, table_ERR2` is entered, meaning that the two tables `table_ERR1` and `table_ERR2` are used to record the error messages.

**Return fastload error**

Select this check box to specify the exit code number to indicate the point at which an error message should display in the console.

**ERRLIMIT**

Enter the limit number of errors detected during the loading phase. Processing stops when the limit is reached. The default error limit value is 1000000.

For more information, see *Teradata FastLoad Reference* documentation.
### tStatCatcher Statistics

<table>
<thead>
<tr>
<th>Usage</th>
<th>tStatCatcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

### Usage

| Usage rule | This component offers the flexibility of the DB query and covers all possible SQL queries. |

## Related scenario

For related topic, see *Scenario: Inserting data into a Teradata database table* on page 3928.
**tTeradataInput**

Executes a DB query with a strictly defined order which must correspond to the schema definition.
tTeradataInput reads a database and extracts fields based on a query. It passes on the field list to the next component via a Main row link.

**tTeradataInput Standard properties**

These properties are used to configure tTeradataInput running in the Standard Job framework.
The Standard tTeradataInput component belongs to the Databases family.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

**Note:**
When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <em>line</em> when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
<td></td>
</tr>
<tr>
<td>Click <strong>Edit schema</strong> to make changes to the schema. If the current schema is of the <strong>Repository</strong> type, three options are available:</td>
<td></td>
</tr>
<tr>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
<td></td>
</tr>
<tr>
<td><strong>Table name</strong></td>
<td>Browse to, or enter the name of the table to be used.</td>
</tr>
<tr>
<td><strong>Query type and Query</strong></td>
<td>Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition.</td>
</tr>
</tbody>
</table>
Advanced settings

<table>
<thead>
<tr>
<th>Advanced settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional JDBC parameters</strong></td>
<td>Specify additional connection properties in the existing DB connection, to allow specific character set support. E.G.: CHARSET=KANJISJIS_OS to get support of Japanese characters.</td>
</tr>
<tr>
<td><strong>Trim all the String/Char columns</strong></td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</td>
</tr>
<tr>
<td><strong>Trim column</strong></td>
<td>Remove leading and trailing whitespace from defined columns.</td>
</tr>
<tr>
<td><strong>Query band</strong></td>
<td>Select this check box to use the Teradata Query Banding feature to add metadata to the query to be processed, such as the user running the query. This can help you, for example, identify the origin of this query.</td>
</tr>
<tr>
<td></td>
<td>Once selecting the check box, the Query Band parameters table is displayed, in which you need to enter the metadata information to be added. This information takes the form of key/value pairs, for example, DpiID in the Key column and Finance in the Value column.</td>
</tr>
<tr>
<td></td>
<td>This check box actually generates the SET QUERY_BAND FOR SESSION statement with the key/value pairs declared in the Query Band parameters table. For further information about this statement, see <a href="http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/SQL_Reference/B035_1144_111A/End_Logging-Syntax.027.143.html">http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/SQL_Reference/B035_1144_111A/End_Logging-Syntax.027.143.html</a>.</td>
</tr>
<tr>
<td></td>
<td>This check box is not available when you have selected the Using an existing connection check box. In this situation, if you need to use the Query Band feature, set it in the Advanced settings tab of the Teradata connection component to be used.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Global Variables</strong></td>
<td><strong>NB_LINE</strong>: the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>QUERY</strong>: the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
</tbody>
</table>
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component covers all possible SQL queries for Teradata databases.</th>
</tr>
</thead>
</table>

Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Related scenarios

For related scenarios, see:

- Scenario: Loading data into a Teradata database on page 3913.
- Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
tTeradataMultiLoad

Executes a database query according to a strict order which must be the same as the one in the schema.
tTeradataMultiLoad reads a database and extracts fields using queries. The retrieved list of fields is then transferred to the next component, using a connection flow (Main row).

**tTeradataMultiLoad Standard properties**

These properties are used to configure tTeradataMultiLoad running in the Standard Job framework.
The Standard tTeradataMultiLoad component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property type</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Built-in</strong></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><strong>Execution platform</strong></td>
<td>Select the Operating System type you use.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Host name or IP address of the database server.</td>
</tr>
<tr>
<td><strong>Database name</strong></td>
<td>Database name.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong></td>
<td>Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Script generated folder</th>
<th>Browse your directory and select the destination of the file which will be created.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action to data</strong></td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Make changes to existing entries</td>
</tr>
<tr>
<td></td>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Remove entries corresponding to the input flow.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning</strong>: It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking <strong>Edit Schema</strong> and selecting the check box(es) next to the column(s) you want to set as primary key(s).</td>
</tr>
<tr>
<td><strong>Where condition in case Delete</strong></td>
<td>Type in a condition, which, once verified, will delete the row. This field appears only when <strong>Delete</strong> is selected from the <strong>Action to data</strong> drop-down list.</td>
</tr>
<tr>
<td><strong>Load file</strong></td>
<td>Browse your directory and select the file from which you want to load data.</td>
</tr>
<tr>
<td><strong>Field separator</strong></td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td><strong>Error file</strong></td>
<td>Browse your directory and select the destination of the file where the error messages will be recorded.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| Define Log table | This check box is selected to define a log table you want to use in place of the default one that is the database |
table you defined in Basic settings. The syntax required to define the log table is `databasename.logtablename`.

BEGIN LOAD
This field allows you to define your BEGIN LOAD command to initiate or restart a load task. You can specify the number of sessions to use, the error limit, any other parameters needed to execute the task.

For more information, see *Teradata MultiLoad Reference* documentation.

Return mload error
Select this check box to specify the exit code number to indicate the point at which an error message should display in the console.

Define character set
Specify the character encoding you need use for your system.

`tStat Catcher Statistics`
Select this check box to collect log data at the component level.

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th><code>EXIT_VALUE</code>: the process exit code. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>ERROR_MESSAGE</code>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

**Usage**

| Usage rule | This component offers the flexibility of the DB query and covers all possible SQL queries. |

**Related scenario**

For related topic, see *Scenario: Inserting data into a Teradata database table* on page 3928.
tTeradataOutput

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the job.

tTeradataOutput writes, updates, makes changes or suppresses entries in a database.

tTeradataOutput Standard properties

These properties are used to configure tTeradataOutput running in the Standard Job framework.

The Standard tTeradataOutput component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-In or Repository.</td>
</tr>
<tr>
<td>Built-In: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td>Repository: Select the repository file where the properties are stored.</td>
<td></td>
</tr>
</tbody>
</table>

Use an existing connection

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

Note:

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

Click this icon to open a database connection wizard and store the database connection parameters you set in the component **Basic settings** view.

For more information about setting up and storing database connection parameters, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Name of the database</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Note:" /></td>
<td>The <strong>Action on table</strong> list will not be available if you select the Enable parallel execution check box in the <strong>Advanced settings</strong> view.</td>
</tr>
<tr>
<td>Note:</td>
<td>On the table defined, you can perform one of the following operations:</td>
</tr>
<tr>
<td><strong>None</strong></td>
<td>No operation is carried out.</td>
</tr>
<tr>
<td><strong>Drop and create a table</strong></td>
<td>The table is removed and created again.</td>
</tr>
<tr>
<td><strong>Create a table</strong></td>
<td>The table does not exist and gets created.</td>
</tr>
<tr>
<td><strong>Create a table if not exists</strong></td>
<td>The table is created if it does not exist.</td>
</tr>
<tr>
<td><strong>Drop a table if exists and create</strong></td>
<td>The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td><strong>Clear a table</strong></td>
<td>The table content is deleted.</td>
</tr>
<tr>
<td><strong>Create</strong></td>
<td>This is not visible by default, until you choose to create a table from the <strong>Action on table</strong> drop-down list. The table to be created may be:</td>
</tr>
<tr>
<td>- <strong>SET TABLE</strong></td>
<td>tables which do not allow to duplicate</td>
</tr>
<tr>
<td>- <strong>MULTI SET TABLE</strong></td>
<td>tables allowing duplicate rows.</td>
</tr>
<tr>
<td><strong>Action on data</strong></td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td><strong>Insert</strong></td>
<td>Add new entries to the table. If duplicates are found, job stops.</td>
</tr>
<tr>
<td><strong>Update</strong></td>
<td>Make changes to existing entries.</td>
</tr>
</tbody>
</table>
**Insert or update:** Insert a new record. If the record with the given reference already exists, an update would be made.

**Update or insert:** Update the record with the given reference. If the record does not exist, a new record would be inserted.

**Delete:** Remove entries corresponding to the input flow.

**Warning:**

*It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the Update and Delete operations. To do that: Select the **Use field options** check box and then in the **Key in update column**, select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the **Key in delete column** for the Delete operation.*

### Schema and Edit schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

**Built-In:** You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository:** You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema:** choose this option to view the schema only.
- **Change to built-in property:** choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection:** choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate
the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Die on error</th>
<th>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row &gt; Rejects link.</th>
</tr>
</thead>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>Additional JDBC parameters</th>
<th>Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the Use an existing connection check box in the Basic settings. This is intended to allow specific character set support. E.G.: CHARSET=KANJISJIS_OS to get support of Japanese characters.</th>
</tr>
</thead>
</table>

> **Note:**
> 
> You can press Ctrl+Space to access a list of predefined global variables.

<table>
<thead>
<tr>
<th>Commit every</th>
<th>Enter the number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at execution. This option is not available if you have selected the Use an existing connection check box in the Basic settings view.</th>
</tr>
</thead>
</table>

> **Note:**
> 
> If you have selected Drop and create table, Create table, Create table if does not exist or Drop table if exists and create from the Action on table list in the Basic settings view, you need to enter 0 in this field to ensure the validity of the SQL statements. For more information about the validity of the SQL statements in Teradata Database, see teradata.

<table>
<thead>
<tr>
<th>Additional Columns</th>
<th>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</th>
</tr>
</thead>
</table>

**Name:** Type in the name of the schema column to be altered or inserted as new column

**SQL expression:** Type in the SQL statement to be executed in order to alter or insert the relevant column data.

**Position:** Select Before, Replace or After following the action to be performed on the reference column.
**Reference column:** Type in a column of reference that the **tDBOutput** can use to place or replace the new or altered column.

**Query band**

Select this check box to use the Teradata Query Banding feature to add metadata to the query to be processed, such as the user running the query. This can help you, for example, identify the origin of this query.

Once selecting the check box, the **Query Band parameters** table is displayed, in which you need to enter the metadata information to be added. This information takes the form of key/value pairs, for example, **DpID** in the **Key** column and **Finance** in the **Value** column.

This check box actually generates the SET QUERY_BAND FOR SESSION statement with the key/value pairs declared in the **Query Band parameters** table. For further information about this statement, see [http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/SQL_Reference/B035_1144_111A/End_Logging-Syntax.027.143.html](http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/SQL_Reference/B035_1144_111A/End_Logging-Syntax.027.143.html).

This check box is not available when you have selected the **Using an existing connection** check box. In this situation, if you need to use the Query Band feature, set it in the **Advanced settings** tab of the Teradata connection component to be used.

**Use field options**

Select this check box to customize a request, especially when there is double action on data.

**Enable debug mode**

Select this check box to display each step during processing entries in a database.

**tStatCatcher Statistics**

Select this check box to collect log data at the component level.

**Use Batch**

Select this check box to activate the batch mode for data processing.

**Batch Size**

Specify the number of records to be processed in each batch.

This field appears only when the **Use batch mode** check box is selected.

**Global Variables**

**Global Variables**

**NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

**NB_LINE_UPDATED**: the number of rows updated. This is an After variable and it returns an integer.

**NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.

**NB_LINE_DELETED**: the number of rows deleted. This is an After variable and it returns an integer.
NB_LINE_REJECTED: the number of rows rejected. This is an After variable and it returns an integer.

ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see Talend Studio User Guide.

Usage

Usage rule

This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.

This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a Teradata database. It also allows you to create a reject flow using a Row > Rejects link to filter data in error.

Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.

For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add
all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

**Related scenarios**

For related topics, see:

- **Scenario: Inserting a column and altering data using tMysqlOutput** on page 2521.
**tTeradataRollback**

Cancels the transaction commit in the Teradata database.

**tTeradataRollback Standard properties**

These properties are used to configure tTeradataRollback running in the Standard Job framework.

The Standard tTeradataRollback component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>TeradataConnection</strong> component in the list if more than one connection are planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tTeradata* components, especially with the **tTeradataConnection** and **tTeradataCommit** components. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable. |
For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenario

For tTeradataRollback related scenario, see Scenario: Rollback from inserting data in mother/daughter tables on page 2480.
tTeradataRow

Acts on the actual DB structure or on the data (although without handling data).
tTeradataRow is the specific component for this database query. It executes the SQL query stated onto the specified database. The row suffix means the component implements a flow in the job design although it doesn’t provide output.
The SQLBuilder tool helps you write easily your SQL statements.

### tTeradataRow Standard properties

These properties are used to configure tTeradataRow running in the Standard Job framework.
The Standard tTeradataRow component belongs to the Databases family.
The component in this framework is generally available.

#### Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

#### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td></td>
<td>Note: When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>Database server IP address</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Schema and Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong></td>
<td>You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Click **Edit schema** to make changes to the schema. If the current schema is of the Repository type, three options are available:
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th><strong>Query type</strong></th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong></td>
<td>Fill in manually the query statement or build it graphically using SQLBuilder</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the relevant query stored in the Repository. The Query field gets accordingly filled in.</td>
</tr>
</tbody>
</table>

| **Query** | Enter your DB query paying particularly attention to properly sequence the fields in order to match the schema definition. |
| **Commit every** | Number of rows to be completed before committing batches of rows together into the DB. This option |
| **Die on error** | This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a **Row > Rejects** link. |
| **Advanced settings** |  |
| **Additional JDBC parameters** | Specify additional connection properties for the DB connection you are creating. This option is not available if you have selected the **Use an existing connection** check box in the **Basic settings**.  
This is intended to allow specific character set support. E.G.: CHARSET=KANJISJIS_OS to get support of Japanese characters. |
| **Propagate QUERY’s recordset** | Select this check box to insert the result of the query into a COLUMN of the current flow. Select this column from the **use column** list.  
**Note:**  
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of **Object** and this component is usually followed by **tParseRecordSet**. |
| **Use PreparedStatement** | Select this check box if you want to query the database using a PreparedStatement. In the **Set PreparedStatement Parameter** table, define the parameters represented by “?” in the SQL instruction of the **Query** field in the **Basic Settings** tab.  
**Parameter Index**: Enter the parameter position in the SQL instruction.  
**Parameter Type**: Enter the parameter type.  
**Parameter Value**: Enter the parameter value.  
**Note:**  
This option is very useful if you need to execute the same query several times. Performance levels are increased |
| **Commit every** | Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions. |
| **Query band** | Select this check box to use the Teradata Query Banding feature to add metadata to the query to be processed, such as the user running the query. This can help you, for example, identify the origin of this query. |
Once selecting the check box, the **Query Band parameters** table is displayed, in which you need to enter the metadata information to be added. This information takes the form of key/value pairs, for example, **DpID** in the **Key** column and **Finance** in the **Value** column.

This check box actually generates the SET QUERY_BAND FOR SESSION statement with the key/value pairs declared in the **Query Band parameters** table. For further information about this statement, see [http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/SQL_Reference/B035_1144_111A/End_Logging-Syntax.027.143.html](http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/SQL_Reference/B035_1144_111A/End_Logging-Syntax.027.143.html).

This check box is not available when you have selected the **Using an existing connection** check box. In this situation, if you need to use the Query Band feature, set it in the **Advanced settings** tab of the Teradata connection component to be used.

<table>
<thead>
<tr>
<th>tStat Catcher Statistics</th>
<th>Select this check box to collect log data at the component level.</th>
</tr>
</thead>
</table>

**Global Variables**

| Global Variables | **QUERY**: the query statement being processed. This is a Flow variable and it returns a string. **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see **Talend Studio User Guide**. |

**Usage**

| Usage rule | This component offers the flexibility of the DB query and covers all possible SQL queries. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, |
when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Limitation**

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the **Install** button on the **Component** tab view. You can also find out and add all missing JARs easily on the **Modules** tab in the **Integration** perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

### Related scenarios

For related scenarios, see:

- **Scenario: Loading data into a Teradata database** on page 3913.
- **Scenario: Combining two flows for selective output** on page 2558.
- **Procedure** on page 625.
- **Scenario: Removing and regenerating a MySQL table index** on page 2552.
tTeradataSCD

Addresses Slowly Changing Dimension needs, reading regularly a source of data and logging the changes into a dedicated SCD table.

TTeradataSCD reflects and tracks changes in a dedicated Teradata SCD table.

**tTeradataSCD Standard properties**

These properties are used to configure tTeradataSCD running in the Standard Job framework.

The Standard tTeradataSCD component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td></td>
</tr>
<tr>
<td>Property Type</td>
<td>Either <strong>Built-In</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-In</strong></td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong></td>
<td>Select the repository file in which the properties are stored. The database connection related fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Type in the IP address or hostname of the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database</strong></td>
<td>Type in the name of the database you want to use.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>Type in the database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Type in the name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
</tbody>
</table>
| **Action on table** | On the table defined, you can perform one of the following operations:  
• **None**: No operation is carried out.  
• **Create table**: The table does not exist and gets created.  
• **Create table if not exists**: The table is created if it does not exist. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields. |
| **Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*. |
| **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*. |
| Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
• **View schema**: choose this option to view the schema only.  
• **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
• **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window. |
### SCD Editor

The SCD editor helps to build and configure the data flow for slowly changing dimension outputs.

For more information, see [SCD management methodology](#) on page 2566.

<table>
<thead>
<tr>
<th><strong>Use memory saving mode</strong></th>
<th>Select this check box to maximize system performance.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source keys include Null</strong></td>
<td>Select this check box to allow the source key columns to have Null values.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning:</strong> Special attention should be paid to the uniqueness of the source key(s) values when this option is selected.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>Select the check box to stop the execution of the Job when an error occurs.</td>
</tr>
<tr>
<td></td>
<td>Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row &gt; Reject link.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Additional JDBC Parameters</strong></th>
<th>Specify additional connection properties for the database connection you are creating.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This field is not visible when the Use an existing connection check box is selected.</td>
</tr>
<tr>
<td><strong>Debug mode</strong></td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th><strong>Global Variables</strong></th>
<th><strong>NB_LINE_UPDATED</strong>: the number of rows updated. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NB_LINE_INSERTED</strong>: the number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
</tbody>
</table>

---

3901
Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as an Output component. It requires an Input component and a Row &gt; Main link as input.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>This component does not support using SCD type 0 together with other SCD types.</td>
</tr>
</tbody>
</table>

Related scenario

For a similar scenario using MySQL database, see Tracking data changes using Slowly Changing Dimensions (type 0 through type 3) on page 2569.
tTeradataSCDELT

Addresses Slowly Changing Dimension needs through SQL queries (server-side processing mode), and logs the changes into a dedicated Teradata SCD table.
tTeradataSCDELT reflects and tracks changes in a dedicated Teradata SCD table.

**tTeradataSCDELT Standard properties**

These properties are used to configure tTeradataSCDELT running in the Standard Job framework.
The Standard tTeradataSCDELT component belongs to the Business Intelligence and the Databases families.
The component in this framework is generally available.

**Note:**
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
</table>
| **Property Type** | Either **Built-In** or **Repository**.  
  - **Built-In**: No property data stored centrally.  
  - **Repository**: Select the repository file in which the properties are stored. The database connection fields that follow are completed automatically using the data retrieved. |

<table>
<thead>
<tr>
<th>Use an existing connection</th>
<th>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</th>
</tr>
</thead>
</table>
| **Note:** | When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:  
  1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.  
  2. In the child level, use a dedicated connection component to read that registered database connection. |
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide.*

<table>
<thead>
<tr>
<th>Host</th>
<th>Enter the IP address or hostname of the database server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Enter the name of the database you want to use.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Enter the user authentication data to access the database. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Source table</td>
<td>Enter the name of the input Teradata SCD table.</td>
</tr>
<tr>
<td>Table</td>
<td>Enter the name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
</tbody>
</table>
| Action on table | Select one of the following operations to be performed on the table defined:  
• **None**: No action carried out on the table.  
• **Create table**: A new table gets created.  
• **Drop and create table**: The table is removed and created again.  
• **Create table if not exists**: A table gets created if it does not exist.  
• **Drop table if exists and create**: The table is removed if it already exists and created again.  
• **Clear table**: The table content is deleted. You have the possibility to rollback the operation.  
• **Truncate table**: The table content is deleted. You do not have the possibility to rollback the operation. |
| Create | Select the type of the table to be created:  
• **SET TABLE**: creates a SET table that does not allow duplicate rows to be inserted.  
• **MULTISET TABLE**: creates a MULTISET table that allows duplicate rows to be inserted.  
This list is available only when Create table, Drop and create table, Create table if not exists, or Drop table if exists and create is selected from the Action on table drop-down list. |
| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.  
• **Built-In**: The schema is created and stored locally for this component only. Related topic: see *Talend Studio User Guide.* |
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

<table>
<thead>
<tr>
<th>Surrogate key</th>
<th>Select the surrogate key column from the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation</td>
<td>Select the method to be used for the surrogate key generation. For more information about the creation methods, see <a href="#">SCD management methodology</a> on page 2566.</td>
</tr>
<tr>
<td>Source Keys</td>
<td>Select one or more columns to be used as keys, to ensure the unicity of the incoming data.</td>
</tr>
<tr>
<td>Use SCD type 1 fields</td>
<td>Select this check box to use SCD Type 1 if tracking changes is not necessary. SCD Type 1 should be used for typos corrections for example.</td>
</tr>
<tr>
<td>SCD type 1 fields</td>
<td>Click the [+] button to add as many rows as needed, and in each row select an input schema column that will be checked for Type 1 changes. This table is available only when the <strong>Use SCD type 1 fields</strong> check box is selected.</td>
</tr>
<tr>
<td>Use SCD type 2 fields</td>
<td>Select this check box to use SCD Type 2 if changes need to be tracked down. SCD Type 2 should be used to trace updates for example.</td>
</tr>
<tr>
<td>SCD type 2 fields</td>
<td>Click the [+] button to add as many rows as needed, and in each row select an input schema column that will be checked for Type 2 changes. This table is available only when the <strong>Use SCD type 2 fields</strong> check box is selected.</td>
</tr>
<tr>
<td>Start date</td>
<td>Select an input schema column to add it to your SCD schema to hold the start date value for the record. This list is available only when the <strong>Use SCD type 2 fields</strong> check box is selected.</td>
</tr>
</tbody>
</table>
### End date
Select an input schema column to add it to your SCD schema to hold the end date value for the record. When the record is currently active, the End Date column shows a null value, or you can select Fixed year value and fill it in with a fictive year to avoid having a null value in the End Date field.

This list is available only when the **Use SCD type 2 fields** check box is selected.

### Log active status
Select this check box and from the Active field list displayed select an input schema column to add it to your SCD schema to hold the true or false status value, which helps to easily spot the active record.

This check box is available only when the **Use SCD type 2 fields** check box is selected.

### Log versions
Select this check box and from the Version field list displayed select an input schema column to add it to your SCD schema to hold the version number of the record.

This check box is available only when the **Use SCD type 2 fields** check box is selected.

### Advanced settings

| **Additional JDBC Parameters** | Specify additional connection properties for the database connection you are creating.  
This field is not available if the **Use an existing connection** check box is selected. |
|--------------------------------|--------------------------------------------------------------------------------------------------|

<table>
<thead>
<tr>
<th><strong>Source fields value include Null</strong></th>
<th>Select this check box to allow the source columns to have Null values. The source columns here refer to the fields defined in the SCD type 1 fields and SCD type 2 fields tables.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>tStatCatcher Statistics</strong></th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

### Global Variables

| **Global Variables** | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component is used as an output component and it always needs an incoming link.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic settings</td>
<td>Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

| Limitation | This component requires installation of its related jar files. |

### Related scenario

For related topics, see *tMysqlSCD on page 2563 and Tracking data changes using Slowly Changing Dimensions (type 0 through type 3) on page 2569.*
tTeradataTPTExec

Offers high performance in inserting data from an existing file to a table in a Teradata database.

As the combination of tTeradataFastLoad, tTeradataMultiLoad, tTeradataTPump, and tTeradataFastExport, tTeradataTPTExec loads the data from an existing file to a Teradata database.

**tTeradataTPTExec Standard properties**

These properties are used to configure tTeradataTPTExec running in the Standard Job framework.

The Standard tTeradataTPTExec component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-In: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The database connection related fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Execution platform</td>
<td>Select the Operating System that will be used to run the Job, either Windows or Unix.</td>
</tr>
<tr>
<td>TDPID</td>
<td>Specify the Teradata director program identifier. It can be either the name or the IP address of the Teradata database system being accessed.</td>
</tr>
<tr>
<td>Database name</td>
<td>Specify the name of the Teradata database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Specify the username and the password for the Teradata database authentication. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. Note: Avoid using a Teradata database keyword as a Db Column name in the schema. If you have to, be sure to enclose the column name in a pair of &quot;. For example, when the keyword id is used as a database column name, the Db Column should be filled with &quot;id&quot;.</td>
</tr>
</tbody>
</table>
### Consumer Operator

Select a consumer operator from the drop-down list.
- **Load**: writes data into an empty Teradata table using the Teradata FastLoad utility protocol.
- **Inserter**: inserts data into Teradata tables with SQL sessions.
- **Update**: performs Insert, Update, or Delete operations using the Teradata MultiLoad utility protocol.
- **Stream**: loads data continuously into Teradata tables using the Teradata TPump utility.

For more information about Teradata consumer operators, see [http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/Load_and_UnloadUtilities/B035_2445_071A/2445ch02.109.18.html](http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/Load_and_UnloadUtilities/B035_2445_071A/2445ch02.109.18.html).

### Action On Data

Select an action to be performed on the data from the drop-down list.
- **Insert**: inserts new records to the table. If duplicates are found, the Job stops.
- **Update**: updates existing records in the table.
- **InsertOrUpdate**: inserts new records or updates existing records based on the given reference key.
- **Delete**: removes records corresponding to the input flow.

**Note:**
You must specify at least one column as the primary key on which the **Update** or **Delete** actions can be performed.
Delete operation is based. You can do that by clicking the `[...]` button next to Edit schema and selecting the check box(es) next to the column(s) you want to set as primary key(s).

This list field appears only when the Update or Stream operator is selected from the Consumer Operator drop-down list.

<table>
<thead>
<tr>
<th>Producer Operator</th>
<th>Select a producer operator from the drop-down list. Currently, only the DataConnector operator is supported. <strong>DataConnector</strong>: accesses files either directly or through an access module, and then writes it to the data stream. For more information about Teradata producer operators, see <a href="http://www.info.teradata.com/HTMLPubs/DB_TTU_13_10/index.html#page/Load_and_UnloadUtilities/B035_2445_020A/2445ch02.04.16.html">http://www.info.teradata.com/HTMLPubs/DB_TTU_13_10/index.html#page/Load_and_UnloadUtilities/B035_2445_020A/2445ch02.04.16.html</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Specify the name of the table to be written into the Teradata database. Note that only one table can be written at a time.</td>
</tr>
<tr>
<td>Script generated folder</td>
<td>Specify the directory under which the Teradata Parallel Transporter script file will be created during the Job execution. This script file will be deleted at the end of the Job execution.</td>
</tr>
<tr>
<td>Load file</td>
<td>Specify the file holding the data to be loaded into the Teradata database.</td>
</tr>
<tr>
<td>Error file</td>
<td>Specify the file in which log messages will be recorded.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Field separator</th>
<th>Character, string or regular expression to separate fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define Log table</td>
<td>Select this check box to specify a log table so that log messages recorded in the log file will also be written into the log table.</td>
</tr>
<tr>
<td>Set Script Parameters</td>
<td>Select this check box to define script parameters, which will be used when generating a script during the Job execution. If you do not specify them manually, the system will use their default values.</td>
</tr>
<tr>
<td>Load Operator</td>
<td>Specify the load operator. This field appears only when the Set Script Parameters check box is selected.</td>
</tr>
<tr>
<td>Data Connector</td>
<td>Specify the data connector. This field appears only when the Set Script Parameters check box is selected.</td>
</tr>
<tr>
<td>Job Name</td>
<td>Specify the name of a Teradata Parallel Transporter Job which is defined using the Teradata <code>tbuild</code> command.</td>
</tr>
</tbody>
</table>

This field appears only when the **Set Script Parameters** check box is selected.

<table>
<thead>
<tr>
<th>Layout Name (schema)</th>
<th>Specify a schema for the data to be loaded. This field appears only when the <strong>Set Script Parameters</strong> check box is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return mload error</td>
<td>Select this check box to specify the exit code number to indicate the point at which an error message should display in the console.</td>
</tr>
<tr>
<td>Define character set</td>
<td>Select this check box to specify the character encoding to be used in your system.</td>
</tr>
</tbody>
</table>

**Apply TPT consumer operator optional attributes**

Select this check box to define optional attribute(s) for the consumer operator that you have selected.


**Optional attributes**

Click the `[+]` button below the table to add as many rows as needed, each row for an optional attribute, and set the following two parameters for each attribute:

- **Name**: click the cell and select an optional attribute from the drop-down list. The list of optional attributes may vary depending on the consumer operator you have selected. For details about supported optional attributes for each consumer operator, see [Supported optional attributes for each consumer operator on page 3912](http://developer.teradata.com/sites/all/files/documentation/linked_docs/2436020A_TPT-Reference-13.10.pdf).

- **Value**: enter the value for the corresponding optional attribute.


This table appears only when the **Apply TPT consumer operator optional attributes** check box is selected.

**tStatCatcher Statistics**

Select this check box to collect the log data at the component level.

**Global Variables**

**Global Variables**

**EXIT_VALUE**: the process exit code. This is an After variable and it returns an integer.
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see **Talend Studio User Guide**.

## Usage

### Usage rule

Used as a single-component Job or Sub-Job, this component offers high performance in inserting data from an existing file to a table in the Teradata database. For further information about the usage of this component, see http://developer.teradata.com/sites/all/files/documentation/linked_docs/html/online/B035-2436-088A/wwhelp/wwhipml/js/html/wwhelp.htm.

### Limitation

The Teradata client tool with the Teradata Parallel Transporter Base and Teradata Parallel Transporter Stream features needs to be installed on the machine where Jobs using this component are executed.

## Supported optional attributes for each consumer operator

This section lists all supported optional attributes for each consumer operator.

### The Load operator

BufferSize, ErrorLimit, MaxSessions, MinSessions, TenacityHours, TenacitySleep, AccountId, DataEncryption, DateForm, ErrorTable1, ErrorTable2, LogonMech, LogonMechData, NotifyExit, NotifyExitIsDLL, NotifyLevel, NotifyMethod, LogSQL, NotifyString, PauseAcq, PrivateLogName, QueryBandSessInfo, WildcardInsert, WorkingDatabase, TraceLevel.

### The Inserter operator

AccountId, DataEncryption, DateForm, LogonMech, LogonMechData, LogSQL, PrivateLogName, QueryBandSessInfo, ReplicationOverride, TraceLevel, WorkingDatabase.

### The Update operator

PrivateLogName, BufferSize, ErrorLimit, MaxSessions, MinSessions, TenacityHours, TenacitySleep, AccountId, AmpCheck, DataEncryption, DateForm, DeleteTask, DropErrorTable, DropLogTable, DropWorkTable, ErrorTable1, ErrorTable2, LogonMech, LogonMechData, LogSQL, NotifyExit, NotifyExitIsDLL, NotifyLevel, NotifyMethod, NotifyString, PauseAcq, QueryBandSessInfo, QueueErrorTable, WorkingDatabase, WorkTable, TraceLevel.
**The Stream operator**

Buffers, ErrorLimit, MaxSessions, MinSessions, Pack, Rate, Periodicity, TenacityHours, TenacitySleep, AccountId, AppendErrorTable, ArraySupport, DataEncryption, DateForm, DropErrorTable, DropMacro, ErrorTable, LogonMech, LogonMechData, MacroDatabase, OperatorCommandID, NotifyExit, NotifyExitsDLL, NotifyLevel, NotifyMethod, LogSQL, NotifyString, PackMaximum, PrivateLogName, QueryBandSessInfo, QueueErrorTable, ReplicationOverride, Robust, WorkingDatabase, TraceLevel.

**Scenario: Loading data into a Teradata database**

This scenario describes a Job that creates a new Teradata database table, writes data into a delimited file, then loads the data from the file into this table, and finally retrieves the data from the table and displays it on the console.

![](image)

**Dropping and linking the components**

**Procedure**

1. Create a new Job and add the following components by typing their names in the design workspace or dropping them from the Palette: a `tTeradataRow` component, a `tFixedFlowInput` component, a `tFileOutputDelimited` component, a `tTeradataTPTExec` component, a `tTeradataInput` component, and a `tLogRow` component.

2. Connect `tFixedFlowInput` to `tFileOutputDelimited` using a `Row > Main` connection.

3. Do the same to connect `tTeradataInput` to `tLogRow`. 
4. Connect tTeradataRow to tFixedFlowInput using a Trigger > On Subjob Ok connection.
5. Do the same to connect tFixedFlowInput to tTeradataTPTExec and tTeradataTPTExec to tTeradataInput.

Configuring the components

Creating a new Teradata database table

Procedure

1. Double-click tTeradataRow to open its Basic settings view.

![tTeradataRow_1](image)

2. Fill in the Host, Database, Username, and Password fields with your Teradata database connection details.

3. In the Query field, enter the following SQL statement to create a new table named person with three columns id, name, sex.

```sql
CREATE SET TABLE samples.person,
FALLBACK,
NO BEFORE JOURNAL,
NO AFTER JOURNAL
(
    id INTEGER NOT NULL,
    name VARCHAR(50),
    sex VARCHAR(20)
)
UNIQUE PRIMARY INDEX (id)
```

Preparing the source data

Procedure

1. Double-click tFixedFlowInput to open its Basic settings view.
2. Click the [...] button next to Edit schema to open the schema editor.

3. Click the [+] button to add three columns: id of the integer type, name and sex of the string type.

4. Click OK to close the schema editor and accept the propagation prompted by the pop-up dialog box.

5. In the Mode area, select Use Inline Content (delimited file) and enter the input data in the Content field.

   1;Ford;Male
   2;Rose;Female
   3;Sabrina;Female
   4;Teddy;Male
   5;Kate;Male

6. Double-click tFileOutputDelimited to open its Basic settings view.
7. In the **File Name** field, specify the file into which the input data will be written. In this example, it is `E:/person.csv`.

## Loading the source data into an empty table

### Procedure

1. Double-click **tTeradataTPTExec** to open its **Basic settings** view.

2. Fill in the **TDPID**, **Database name**, **Username**, and **Password** fields with your Teradata database connection details.

3. In the **Table** field, enter the name of the table into which the source data will be loaded. In this example, it is `person`.

4. In the **Script generated folder** field, browse to the directory under which the Teradata Parallel Transporter script file will be created during the Job execution. In this example, it is `E:/`.

5. In the **Load file** field, browse to the file that contains the source data. In this example, it is `E:/person.csv`.

6. In the **Error file** field, specify the file in which log messages will be recorded. In this example, it is `E:/error.log`.
7. Click the [...] button next to **Edit schema** to open the schema editor.

![Schema editor screenshot]

Click the [+] button to add three columns: *id* of the integer type, *name* and *sex* of the string type. Note that *id* and *name* in **Db Column** are enclosed in a pair of `\` since they are Teradata database keywords.

Click **OK** to validate these changes and close the schema editor.

8. Click **Advanced settings** to open its view, and then select the **Apply TPT consumer operator optional attributes** check box and click the [+] button below the **Optional attributes** table to add the following attributes needed: *ErrorLimit*, *ErrorTable1*, *QueryBandSessInfo*, and *TraceLevel*.

![Advanced settings screenshot]

**Note:**

For VARCHAR attributes, enter their values between double quotation marks.
Retrieving data from the Teradata database table

Procedure

1. Double-click **tTeradataInput** to open its **Basic settings** view.

2. In the **Table Name** field, enter the name of the table to read data from. In this example, it is `person`.

3. In the **Query** field, enter the following SQL statement to retrieve data from the table `person`.

   ```sql
   SELECT * FROM samples.person ORDER BY id
   ```

4. Click the `[...]` button next to **Edit schema** to open the schema editor.

5. Click the `[+]` button to add three columns: `id` of the integer type, `name` and `sex` of the string type. Note that `id` and `name` in **Db Column** are enclosed in a pair of `\` since they are Teradata database keywords.

6. Click **OK** to close the schema editor and accept the propagation prompted by the pop-up dialog box.

7. Double-click **tLogRow** to open its **Basic settings** view.
8. In the **Mode** area, select the **Table (print values in cells of a table)** option for a better display of the result.

### Saving and executing the Job

**Procedure**

1. Press **Ctrl + S** to save the Job.
2. Press **F6** to execute the Job.

```
Starting job tTeradataTPTExec_Demo at 10:35 28/09/2014.
[statistics] connecting to socket on port 3553
[statistics] connected
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>
[statistics] disconnected
Job tTeradataTPTExec_Demo ended at 10:35 28/09/2014. [exit code=0]
```

The data written into the specified Teradata database table is displayed on the console.
tTeradataTPTUtility

Writes the incoming data to a file and then loads the data from the file to a Teradata database.

As the combination of tTeradataFastLoad, tTeradataMultiLoad, tTeradataTPump, and tTeradataFastExport, tTeradataTPTUtility writes the incoming data to a file and then loads the data from the file to a Teradata database.

**tTeradataTPTUtility Standard properties**

These properties are used to configure tTeradataTPTUtility running in the Standard Job framework.
The Standard tTeradataTPTUtility component belongs to the Databases family.
The component in this framework is generally available.

**Basic settings Data file**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The <strong>Filename</strong> field is completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Filename**
Specify the file to save the incoming data.

**Append**
Select this check box to append the incoming data in the file specified in the **Filename** field.

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Either Built-In or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Built-In</strong>: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository</strong>: Select the repository file in which the properties are stored. The database connection related fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Execution platform**
Select the Operating System that will be used to run the Job, either Windows or Unix.

**TDPID**
Specify the Teradata director program identifier. It can be either the name or the IP address of the Teradata database system being accessed.

**Database name**
Specify the name of the Teradata database.

**Username and Password**
Specify the username and the password for the Teradata database authentication.
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter
### Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

**Note:**

Avoid using a Teradata database keyword as a **Db Column** name in the schema. If you have to, be sure to enclose the column name in a pair of `\`. For example, when the keyword `id` is used as a database column name, the **Db Column** should be filled with `\id\`.

### Built-In

You create and store the schema locally for this component only. Related topic: see [Talend Studio User Guide](https://www.talend.com/).  

### Repository

You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see [Talend Studio User Guide](https://www.talend.com/).

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

### Consumer Operator

Select a consumer operator from the drop-down list.

- **Load**: writes data into an empty Teradata table using the Teradata FastLoad utility protocol.
- **Inserter**: inserts data into Teradata tables with SQL sessions.
- **Update**: performs Insert, Update, or Delete operations using the Teradata MultiLoad utility protocol.
- **Stream**: loads data continuously into Teradata tables using the Teradata TPump utility.

For more information about Teradata consumer operators, see [http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/Load_and_Unload_Utildies/B035_2445_071A/2445ch02.109.18.html](http://www.info.teradata.com/HTMLPubs/DB_TTU_14_00/index.html#page/Load_and_Unload_Utildies/B035_2445_071A/2445ch02.109.18.html).
### Action On Data

Select an action to be performed on the data from the drop-down list.

- **Insert**: inserts new records to the table. If duplicates are found, the Job stops.
- **Update**: updates existing records in the table.
- **InsertOrUpdate**: inserts new records or updates existing records based on the given reference key.
- **Delete**: removes records corresponding to the input flow.

**Note:**
You must specify at least one column as the primary key on which the **Update** or **Delete** operation is based. You can do that by clicking the [...] button next to **Edit schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s).

This list field appears only when the **Update** or **Stream** operator is selected from the **Consumer Operator** drop-down list.

### Producer Operator

Select a producer operator from the drop-down list. Currently, only the **DataConnector** operator is supported.

**DataConnector**: accesses files either directly or through an access module, and then writes it to the data stream.


### Table

Specify the name of the table to be written into the Teradata database. Note that only one table can be written at a time.

### Script generated folder

Specify the directory under which the Teradata Parallel Transporter script file will be created during the Job execution. This script file will be deleted at the end of the Job execution.

### Error file

Specify the file in which log messages will be recorded.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row separator</strong></td>
<td>Character, string or regular expression to separate rows.</td>
</tr>
<tr>
<td><strong>Field separator</strong></td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td><strong>Include header</strong></td>
<td>Select this check box to include the column header to the file.</td>
</tr>
</tbody>
</table>
### Encoding
Select the encoding from the list or select **Custom** and define it manually. This field is compulsory for database data handling.

### Set Script Parameters
Select this check box to define script parameters, which will be used when generating a script during the Job execution. If you do not specify them manually, the system will use their default values.

### Load Operator
Specify the load operator.
This field appears only when the **Set Script Parameters** check box is selected.

### Data Connector
Specify the data connector.
This field appears only when the **Set Script Parameters** check box is selected.

### Job Name
Specify the name of a Teradata Parallel Transporter Job which is defined using the Teradata `tbuilt` command.
This field appears only when the **Set Script Parameters** check box is selected.

### Layout Name (schema)
Specify a schema for the data to be loaded.
This field appears only when the **Set Script Parameters** check box is selected.

### Define Log table
Select this check box to specify a log table so that log messages recorded in the log file will also be written into the log table.

### Return mload error
Select this check box to specify the exit code number to indicate the point at which an error message should display in the console.

### Define character set
Select this check box to specify the character encoding to be used in your system.

### Apply TPT consumer operator optional attributes
Select this check box to define optional attribute(s) for the consumer operator that you have selected.
For more information about optional attributes of each consumer operator, see [Teradata Parallel Transporter Reference](http://developer.teradata.com/sites/all/files/documentation/linked_docs/2436020A_TPT-Reference-13.10.pdf) guide.

### Optional attributes
Click the `[+]` button below the table to add as many rows as needed, each row for an optional attribute, and set the following two parameters for each attribute:

- **Name**: click the cell and select an optional attribute from the drop-down list. The list of optional attributes may vary depending on the consumer.
operator you have selected. For details about supported optional attributes for each consumer operator, see Supported optional attributes for each consumer operator on page 3912.

- **Value**: enter the value for the corresponding optional attribute.


This table appears only when the **Apply TPT consumer operator optional attributes** check box is selected.

### tStatCatcher Statistics

| tStatCatcher Statistics | Select this check box to collect the log data at the component level. |

### Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>EXIT_VALUE: the process exit code. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Variables</td>
<td><strong>ERROR_MESSAGE</strong>: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the <strong>Die on error</strong> check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td>Global Variables</td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td>Global Variables</td>
<td>To fill up a field or expression with a variable, press <strong>Ctrl + Space</strong> to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td>Global Variables</td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

### Usage

| Usage rule | Preceded by an input component, **tTeradataTPTUtility** writes the incoming data to a file and then loads the data from the file to the Teradata Database. High performance is provided during this process. For further information about the usage of this component, see http://developer.teradata.com/sites/all/files/documentation/linked_docs/html/online/B035-2436-088A/wwwhelp/wwwhelp.htm. |

| Limitation | The Teradata client tool with the Teradata Parallel Transporter Base and Teradata Parallel Transporter Stream features needs to be installed on the machine where Jobs using this component are executed. |
Related scenario

For a related scenario, see Scenario: Loading data into a Teradata database on page 3913.
**tTeradataTPump**

Inserts, updates, or deletes data in the Teradata database with the TPump loading utility which allows near-real-time data to be achieved in the data warehouse.

Use this component particularly for environments where batch windows are shrinking and warehouse maintenance overlaps normal working hours.

**tTeradataTPump Standard properties**

These properties are used to configure tTeradataTPump running in the Standard Job framework.

The Standard tTeradataTPump component belongs to the Databases family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Execution platform**

Select the Operating System type you use.

**Host**

Host name or IP address of the database server.

**Database name**

Database name.

**Username and Password**

DB user authentication data.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.

**Table**

Name of the table to be written. Note that only one table can be written at a time.

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.

**Built-In**

You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

**Repository**

You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.
Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Script generated folder</th>
<th>Browse your directory and select the destination of the file which will be created.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action to data</strong></td>
<td>On the data of the table defined, you can perform:</td>
</tr>
<tr>
<td></td>
<td><strong>Insert</strong>: Add new entries to the table. If duplicates are found, job stops.</td>
</tr>
<tr>
<td></td>
<td><strong>Update</strong>: Make changes to existing entries</td>
</tr>
<tr>
<td></td>
<td><strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
<tr>
<td></td>
<td><strong>Delete</strong>: Remove entries corresponding to the input flow.</td>
</tr>
<tr>
<td><strong>Warning:</strong></td>
<td><em>It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking Edit Schema and selecting the check box(es) next to the column(s) you want to set as primary key(s).</em></td>
</tr>
<tr>
<td><strong>Where condition in case Delete</strong></td>
<td>Type in a condition, which, once verified, will delete the row.</td>
</tr>
<tr>
<td></td>
<td>This field appears only when <strong>Delete</strong> is selected from the <strong>Action to data</strong> drop-down list.</td>
</tr>
<tr>
<td><strong>Load file</strong></td>
<td>Browse your directory and select the file from which you want to load data.</td>
</tr>
<tr>
<td><strong>Field separator</strong></td>
<td>Character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td><strong>Error file</strong></td>
<td>Browse your directory and select the destination of the file where the error messages will be recorded.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **Define Log table**            | This check box is selected to define a log table you want to use in place of the default one that is the database |
BEGIN LOAD

This field allows you to define your BEGIN LOAD command to initiate or restart a TPump task. You can specify the number of sessions to use, the error limit and any other parameters needed to execute the task. The default value is:

SESSIONS 8 PACK 600 ARRAYSUPPORT ON
CHECKPOINT 60 TENACITY 2 ERRLIMIT 1000.

For more information, see Teradata Parallel Data Pump Reference documentation.

Return tpump error

Select this check box to specify the exit code number to indicate the point at which an error message should display in the console.

Define character set

Specify the character encoding you need use for your system.

tStat Catcher Statistics

Select this check box to collect log data at the component level.

Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>EXIT_VALUE: the process exit code. This is an After variable and it returns an integer. ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. For further information about variables, see Talend Studio User Guide.</th>
</tr>
</thead>
</table>

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
</table>

Scenario: Inserting data into a Teradata database table

In this scenario, you create a Job using tTeradataTPump to insert customer data into a Teradata database table and specify the exit code to be displayed in the event of an exception error.

Three components are used in this Job:
• **tRowGenerator**: generates rows as required using random customer data taken from a list.
• **tFileOutputDelimited**: outputs the customer data into a delimited file.
• **tTeradataTPump**: inserts the customer data into the Teradata database table in the Tpump mode.

### Dropping components

**Procedure**

1. Drop the required components: **tRowGenerator**, **tFileOutputDelimited** and **tTeradataTPump** from the Palette onto the design workspace.
2. Link **tRowGenerator** to **tFileOutputDelimited** using a Row > Main connection.
3. Link **tRowGenerator** to **tTeradataTPump** using a Trigger > On SubjobOk connection.

### Configuring the components

**Procedure**

1. Double click **tRowGenerator** to open the [tRowGenerator Editor] window.
   
   In the [tRowGenerator Editor] window, define the data to be generated. For this Job, the schema is composed of two columns: *ID* and *Name*.
Enter the **Number of Rows for RowGenerator** to generate.

2. Double click **tFileOutputDelimited** to define its properties in the **Component** view.

3. Next to **File Name**, browse to the output file or enter a name for the output file to be created.

4. Between double quotation marks, enter the delimiters to be used next to **Row Separator** and **Field Separator**.

5. **Click Edit schema** and check that the schema matches the input schema. If need be, **click Sync Columns**.

6. **Double click tTeradataTPump** to open its **Component** view.

   In the **Basic settings** tab of the **Component** view, define the **tTeradataTPump** parameters. I

7. **Enter the Database name**, **User name** and **Password** in accordance with your database authentication information.

8. Specify the **Table** into which you want to insert the customer data. In this scenario, it is called **mytable**.

9. In the **Script generated folder** field, browse to the folder in which you want to store the script files generated.

10. In the **Load file** field, browse to the file which contains the customer data.
10. In the Error file field, browse to the file in which you want to log the error information.

11. In the Action on data field, select Insert.

**Executing the Job**

**Procedure**

1. Press F6 to execute the Job.
2. The Run view console reads as follows:

```
Running job 'tpump'...
Starting job tpump at 17:01 19/07/2010.
[statistics] connecting to socket on port 3740
[statistics] connected
[statistics] disconnected
Job tpump ended at 17:02 19/07/2010. [exit code=0]
```

3. Double-click the tTeradataTPump component to go back to its Component view.

4. On the Advanced settings tab, select the Return tpump error check box and type in the exit code number to indicate the point at which an error message should be displayed in the console. In this example, enter the number 4 and use the default values for the other parameters.

5. Press F6 to run the Job.

6. The Run view console reads as follows:

```
Running job 'tpump'...
Starting job tpump at 17:56 19/07/2010.
[statistics] connecting to socket on port 3549
[statistics] connected
[statistics] disconnected
Exception in component tTeradataTPump_1
java.lang.RuntimeException: TPump returned exit code 12
at bug.tpump_0_1.tpump.tTeradataTPump_1Process(tpump.java:307)
at bug.tpump_0_1.tpump.runJobInOS(tpump.java:504)
at bug.tpump_0_1.tpump.main(tpump.java:378)
Job tpump ended at 17:56 19/07/2010. [exit code=0]
```

An exception error occurs and TPump returned exit code 12 is displayed. If you need to view detailed information about the exception error, you can open the log file stored in the directory you specified in the Error file field in the Basic settings tab of the Component view.
**tUniqRow**

Ensures data quality of input or output flow in a Job.
tUniqRow compares entries and sorts out duplicate entries from the input flow.

**tUniqRow Standard properties**

These properties are used to configure tUniqRow running in the Standard Job framework.
The Standard tUniqRow component belongs to the Data Quality family.
The component in this framework is generally available.

**Basic settings**

| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.
Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:
- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

| Built-In: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide. |
| Repository: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide. |

| Unique key | In this area, select one or more columns to carry out de duplication on the particular column(s)
- Select the **Key attribute** check box to carry out deduplication on all the columns
- Select the **Case sensitive** check box to differentiate upper case and lower case |
## Advanced settings

<table>
<thead>
<tr>
<th>Only once each duplicated key</th>
<th>Select this check box if you want to have only the first duplicated entry in the column(s) defined as key(s) sent to the output flow for duplicates.</th>
</tr>
</thead>
</table>
| Use of disk (suitable for processing large row set) | Select this check box to enable generating temporary files on the hard disk when processing a large amount of data. This helps to prevent Job execution failure caused by memory overflow. With this check box selected, you need also to define:  
- **Buffer size in memory**: Select the number of rows that can be buffered in the memory before a temporary file is to be generated on the hard disk.  
- **Directory for temp files**: Set the location where the temporary files should be stored.  

**Warning:**  
*Make sure that you specify an existing directory for temporary files; otherwise your Job execution will fail.* |
| Ignore trailing zeros for BigDecimal | Select this check box to ignore trailing zeros for BigDecimal data. |
| tStatCatcher Statistics | Select this check box to gather the job processing metadata at a job level as well as at each component level. |

## Global Variables

| Global Variables | **NB_UNIQUES**: the number of unique rows. This is an After variable and it returns an integer.  
**NB_DUPLICATES**: the number of duplicate rows. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  | **NB_UNIQUES**: the number of unique rows. This is an After variable and it returns an integer.  
**NB_DUPLICATES**: the number of duplicate rows. This is an After variable and it returns an integer.  
**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see *Talend Studio User Guide*. |
**Usage**

| Usage rule | This component handles flow of data therefore it requires input and output, hence is defined as an intermediary step. |

---

**Scenario 1: Deduplicating entries**

In this five-component Job, we will sort entries on an input name list, find out duplicated names, and display the unique names and the duplicated names on the **Run** console.

---

**Setting up the Job**

**Procedure**

1. Drop a **tFileInputDelimited**, a **tSortRow**, a **tUniqRow**, and two **tLogRow** components from the **Palette** to the design workspace, and name the components as shown above.
2. Connect the **tFileInputDelimited** component, the **tSortRow** component, and the **tUniqRow** component using **Row > Main** connections.
3. Connect the **tUniqRow** component and the first **tLogRow** component using a **Main > Uniques** connection.
4. Connect the **tUniqRow** component and the second **tLogRow** component using a **Main > Duplicates** connection.

**Configuring the components**

**Procedure**

1. Double-click the **tFileInputDelimited** component to display its **Basic settings** view.
2. Click the [...] button next to the File Name field to browse to your input file.

3. Define the header and footer rows. In this use case, the first row of the input file is the header row.

4. Click Edit schema to define the schema for this component. In this use case, the input file has five columns: Id, FirstName, LastName, Age, and City. Then click OK to propagate the schema and close the schema editor.

5. Double-click the tSortRow component to display its Basic settings view.

6. To rearrange the entries in the alphabetic order of the names, add two rows in the Criteria table by clicking the plus button, select the FirstName and LastName columns under Schema column, select alpha as the sorting type, and select the sorting order.

7. Double-click the tUniqRow component to display its Basic settings view.
8. In the **Unique key** area, select the columns on which you want deduplication to be carried out. In this use case, you will sort out duplicated names.

9. In the **Basic settings** view of each of the tLogRow components, select the **Table** option to view the Job execution result in table mode.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.
2. Run the Job by pressing **F6** or clicking the **Run** button on the **Run** tab.

The unique names and duplicated names are displayed in different tables on the **Run** console.

---

<table>
<thead>
<tr>
<th>Id</th>
<th>FirstName</th>
<th>LastName</th>
<th>Age</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rutherford</td>
<td>Garfield</td>
<td>74</td>
<td>Honolulu</td>
</tr>
<tr>
<td>11</td>
<td>Rutherford</td>
<td>Roosevelt</td>
<td>66</td>
<td>Oklahoma City</td>
</tr>
<tr>
<td>17</td>
<td>Theodore</td>
<td>Tyler</td>
<td>42</td>
<td>Bismarck</td>
</tr>
<tr>
<td>19</td>
<td>Thomas</td>
<td>Lincoln</td>
<td>44</td>
<td>Augusta</td>
</tr>
<tr>
<td>15</td>
<td>Ulysses</td>
<td>Johnson</td>
<td>37</td>
<td>Austin</td>
</tr>
<tr>
<td>39</td>
<td>Ulysses</td>
<td>Kennedy</td>
<td>27</td>
<td>Phoenix</td>
</tr>
<tr>
<td>30</td>
<td>Ulysses</td>
<td>Pierce</td>
<td>55</td>
<td>Madison</td>
</tr>
<tr>
<td>12</td>
<td>Warren</td>
<td>Jefferson</td>
<td>40</td>
<td>Frankfort</td>
</tr>
<tr>
<td>17</td>
<td>Warren</td>
<td>Johnson</td>
<td>66</td>
<td>Oklahoma City</td>
</tr>
<tr>
<td>18</td>
<td>William</td>
<td>Hoover</td>
<td>76</td>
<td>Columbus</td>
</tr>
<tr>
<td>59</td>
<td>Woodrow</td>
<td>Fillmore</td>
<td>42</td>
<td>Saint Paul</td>
</tr>
<tr>
<td>78</td>
<td>Woodrow</td>
<td>Roosevelt</td>
<td>66</td>
<td>Atlanta</td>
</tr>
<tr>
<td>38</td>
<td>Woodrow</td>
<td>Taft</td>
<td>35</td>
<td>Oklahoma City</td>
</tr>
<tr>
<td>40</td>
<td>Woodrow</td>
<td>Tyler</td>
<td>75</td>
<td>Concord</td>
</tr>
<tr>
<td>16</td>
<td>Zachary</td>
<td>Monroe</td>
<td>35</td>
<td>Richmond</td>
</tr>
<tr>
<td>11</td>
<td>Zachary</td>
<td>Taft</td>
<td>63</td>
<td>Honolulu</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Id</th>
<th>FirstName</th>
<th>LastName</th>
<th>Age</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Calvin</td>
<td>Eisenhower</td>
<td>27</td>
<td>Montgomery</td>
</tr>
<tr>
<td>79</td>
<td>Calvin</td>
<td>Eisenhower</td>
<td>67</td>
<td>Juneau</td>
</tr>
<tr>
<td>14</td>
<td>George</td>
<td>Harrison</td>
<td>74</td>
<td>Harrisburg</td>
</tr>
<tr>
<td>59</td>
<td>Ulysses</td>
<td>Johnson</td>
<td>71</td>
<td>Olympia</td>
</tr>
</tbody>
</table>

---
tUnite

Centralizes data from various and heterogeneous sources.
tUnite merges data from various sources, based on a common schema.
Note that tUnite cannot exist in a data flow loop. For instance, if a data flow goes through several tMap components to generate two flows, they cannot be fed to tUnite.

### tUnite Standard properties

These properties are used to configure tUnite running in the Standard Job framework.
The Standard tUnite component belongs to the Orchestration family.
The component in this framework is generally available.

#### Basic settings

| **Schema and Edit Schema** | A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either Built-in or stored remotely in the Repository. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:
| | • View schema: choose this option to view the schema only.
| | • Change to built-in property: choose this option to change the schema to Built-in for local changes.
| | • Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.
| | Click Sync columns to retrieve the schema from the previous component in the Job.
| Built-in: The schema will be created and stored locally for this component only. Related topic: see Talend Studio User Guide.
| Repository: The schema already exists and is stored in the Repository, hence can be reused in various projects and Job designs. Related topic: see Talend Studio User Guide. |
Advanced settings

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |
| | NB_LINE: the number of rows processed. This is an After variable and it returns an integer. |
| | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
| | To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. |
| | For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | This component is not startable and requires one or several input components and an output component. |

Connections

| Connections | Outgoing links (from this component to another): 
Row: Main. 
Trigger: Run if; On Component Ok; On Component Error |
| | Incoming links (from one component to this one): 
Row: Main; Reject. |
| | For further information regarding connections, see Talend Studio User Guide. |

Scenario: Iterate on files and merge the content

The following Job iterates on a list of files then merges their content and displays the final 2-column content on the console.
Dropping and linking the components

Procedure

1. Drop the following components onto the design workspace: tFileList, tFileInputDelimited, tUnite and tLogRow.
2. Connect the tFileList to the tFileInputDelimited using an Iterate connection and connect the other component using a row main link.

Configuring the components

Procedure

1. In the tFileList Basic settings view, browse to the directory, where the files to merge are stored.

   The files are pretty basic and contain a list of countries and their respective score.

2. In the Case Sensitive field, select Yes to consider the letter case.
3. Select the tFileInputDelimited component, and display this component’s Basic settings view.
4. Fill in the **File Name/Stream** field by using the **Ctrl+Space bar** combination to access the variable completion list, and selecting `tFileList.CURRENT_FILEPATH` from the global variable list to process all files from the directory defined in the `tFileList`.

5. Click the **Edit Schema** button and set manually the 2-column schema to reflect the input files’ content.

For this example, the 2 columns are **Country** and **Points**. They are both **nullable**. The **Country** column is of **String** type and the **Points** column is of **Integer** type.

6. Click **OK** to validate the setting and accept to propagate the schema throughout the Job.

7. Then select the **tUnite** component and display the **Component view**. Notice that the output schema strictly reflects the input schema and is read-only.

8. In the **Basic settings** view of **tLogRow**, select the **Table** option to display properly the output values.

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save your Job.

2. Press **F6**, or click **Run** on the **Run** console to execute the Job.

   The console shows the data from the various files, merged into one single table.
tVectorWiseCommit

Commits a global transaction in one go using a single connection instead of doing so on every row or every batch. This provides a gain in performance.
tVectorWiseCommit validates the data processed in a Job into the connected DB.

tVectorWiseCommit Standard properties

These properties are used to configure tVectorWiseCommit running in the Standard Job framework.
The Standard tVectorWiseCommit component belongs to the Databases family.
The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.
For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the tVectorWiseConnection component from the list if more than one connection is planned for the current job.</td>
</tr>
<tr>
<td>Close connection</td>
<td>This check box is selected by default. It allows you to close the database connection once the commit is done. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

Warning:
If you want to use a Row > Main connection to link tVectorWiseCommit to your Job, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of your first row commit.

Advanced settings

| tStatCatcher Statistics         | Select this check box to collect log data at the component level. |

Usage

| Usage rule                     | This component is more commonly used with other tVectorWise* components, especially with the... |
**Dynamic settings**

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job.

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

**Related scenario**

For a **tVectorWiseCommit** related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476.
tVectorWiseConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

**tVectorWiseConnection Standard properties**

These properties are used to configure tVectorWiseConnection running in the Standard Job framework.

The Standard tVectorWiseConnection component belongs to the Databases and the ELT families.

The component in this framework is generally available.

⚠️ **Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Database</strong></th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: No property data stored centrally.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the repository file where the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
<td></td>
</tr>
<tr>
<td><strong>Server</strong></td>
<td>Database server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>Authentication information of the database user.</td>
</tr>
<tr>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
<td></td>
</tr>
<tr>
<td><strong>Use or register a shared DB Connection</strong></td>
<td>Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the <strong>Shared DB Connection Name</strong> field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.</td>
</tr>
</tbody>
</table>
This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the **tRunJob** component. Using a shared connection together with a **tRunJob** component with either of these two options enabled will cause your Job to fail.

### Advanced settings

<table>
<thead>
<tr>
<th>Advanced setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Auto Commit</strong></td>
<td>Select this check box to commit any changes to the database automatically upon the transaction. With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component. Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage rule</strong></td>
<td>This component is more commonly used with other <strong>tVectorWise</strong> components, especially with the <strong>tVectorWiseCommit</strong> and <strong>tVectorWiseRollback</strong> components.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Limitation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Limitation</strong></td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the <strong>Install</strong> button on the <strong>Component</strong> tab view. You can also find out and add all missing JARs easily on the <strong>Modules</strong> tab in the <strong>Integration</strong> perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

### Related scenario

For a **tVectorWiseConnection** related scenario, see **tMysqlConnection** on page 2475.
tVectorWiseInput

Executes a DB query with a strictly defined order which must correspond to the schema definition. tVectorWiseInput reads a database and extracts fields based on a query. It passes on the field list to the next component via a Main row link.

**tVectorWiseInput Standard properties**

These properties are used to configure tVectorWiseInput running in the Standard Job framework. The Standard tVectorWiseInput component belongs to the Databases family. The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-in:</strong> No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> Select the repository file where Properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component <strong>Basic settings</strong> view. For more information about setting up and storing database connection parameters, see <strong>Talend Studio User Guide</strong>.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td>![Icon]</td>
<td><strong>Note:</strong> When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component</td>
</tr>
</tbody>
</table>
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](#).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Database server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Listening port number of the DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Authentication information of the database user. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Schema and Edit Schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td>Table name</td>
<td>Name of the table to be read.</td>
</tr>
<tr>
<td>Query type and Query</td>
<td>Enter your DB query, ensuring that the field order matches the order in the schema.</td>
</tr>
</tbody>
</table>
**Guess Query**
Click this button to generate a query that corresponds to your table schema in the Query field.

**Guess schema**
Click this button to retrieve the schema from the table.

### Advanced settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trim all the String/Char columns</td>
<td>Select this check box to remove leading and trailing whitespace from all the String/Char columns.</td>
</tr>
<tr>
<td>Trim column</td>
<td>Define columns from which to remove leading and trailing whitespace.</td>
</tr>
<tr>
<td>tStat Catcher Statistics</td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>the number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>QUERY</td>
<td>the query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
</tbody>
</table>

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage rule</td>
<td>This component covers all possible SQL queries for Vertica databases.</td>
</tr>
</tbody>
</table>

| Limitation               | Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com). |
Related scenario

For tVectorWisInput related scenarios, see:

- Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
**tVectorWiseOutput**

Executes the action defined on the table and/or on the data contained in the table, based on the flow incoming from the preceding component in the Job.

tVectorWiseOutput writes, updates, makes changes or suppresses entries in a database.

**tVectorWiseOutput Standard properties**

These properties are used to configure tVectorWiseOutput running in the Standard Job framework.

The Standard tVectorWiseOutput component belongs to the Databases family.

The component in this framework is generally available.

<i>Note:</i>

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td></td>
<td>Built-in: No property data stored centrally.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file in which the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td><img src="icon" alt="icon" /></td>
<td>Click this icon to open a database connection wizard and store the database connection parameters you set in the component Basic settings view. For more information about setting up and storing database connection parameters, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

<i>Note:</i>

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component
which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Host</th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Listening port number of the DB server.</td>
</tr>
<tr>
<td>Database</td>
<td>Name of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>Name of the table to be written. Note that only one table can be written at a time.</td>
</tr>
</tbody>
</table>
| Action on table | On the table defined, you can perform one of the following operations:  
  * None: No operation is carried out.  
  * Drop and create a table: The table is removed and created again.  
  * Create a table: The table does not exist and gets created.  
  * Create a table if not exists: The table is created if it does not exist.  
  * Drop a table if exists and create: The table is removed if it already exists and created again.  
  * Clear a table: The table content is deleted. |
| Action on data  | On the data of the table defined, you can perform:  
  * Insert: Add new entries to the table. If duplicates are found, job stops.  
  * Update: Make changes to existing entries  
  * Insert or update: Insert a new record. If the record with the given reference already exists, an update would be made.  
  * Update or insert: Update the record with the given reference. If the record does not exist, a new record would be inserted.  
  * Delete: Remove entries corresponding to the input flow.  
  
  **Warning:**  
  * It is necessary to specify at least one column as a primary key on which the Update and Delete operations are based. You can do that by clicking...* |
**Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the Advanced settings view where you can simultaneously define primary keys for the Update and Delete operations. To do that: Select the Use field options check box and then in the Key in update column, select the check boxes next to the column names you want to use as a base for the Update operation. Do the same in the Key in delete column for the Delete operation.

### Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

### Built-In

**Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.

### Repository

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

### Die on error

This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a Row > Rejects link.
## Advanced settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Commit every</strong></td>
<td>Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and, above all, better performance at executions.</td>
</tr>
<tr>
<td><strong>Additional Columns</strong></td>
<td>This option is not offered if you create (with or without drop) the DB table. This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing.</td>
</tr>
<tr>
<td></td>
<td><strong>Name:</strong> Type in the name of the schema column to be altered or inserted as new column.</td>
</tr>
<tr>
<td></td>
<td><strong>SQL expression:</strong> Type in the SQL statement to be executed in order to alter or insert the relevant column data.</td>
</tr>
<tr>
<td></td>
<td><strong>Position:</strong> Select Before, Replace or After following the action to be performed on the reference column.</td>
</tr>
<tr>
<td></td>
<td><strong>Reference column:</strong> Type in a column of reference that the tDBOutput can use to place or replace the new or altered column.</td>
</tr>
<tr>
<td><strong>Use field options</strong></td>
<td>Select this check box to customize a request, especially when there is double action on data.</td>
</tr>
<tr>
<td><strong>Enable debug mode</strong></td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td><strong>Support null in “SQL WHERE” statement</strong></td>
<td>Select this check box if you want to deal with the Null values contained in a DB table.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Ensure that the Nullable check box is selected for the corresponding columns in the schema.</td>
</tr>
<tr>
<td><strong>Use Batch</strong></td>
<td>Select this check box to activate the batch mode for data processing.</td>
</tr>
<tr>
<td></td>
<td>This option is only supported with Vectorwise version 2.5 and later, and is only available when you select Insert from the Action on data drop-down list in the Basic settings tab.</td>
</tr>
<tr>
<td><strong>Batch Size</strong></td>
<td>Specify the number of records to be processed in each batch.</td>
</tr>
<tr>
<td></td>
<td>This field appears only when the Use batch mode check box is selected.</td>
</tr>
<tr>
<td><strong>tStat Catcher Statistics</strong></td>
<td>Select this check box to collect log data at the component level.</td>
</tr>
</tbody>
</table>
Global Variables

- **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.
- **NB_LINE_UPDATED**: the number of rows updated. This is an After variable and it returns an integer.
- **NB_LINE_INSERTED**: the number of rows inserted. This is an After variable and it returns an integer.
- **NB_LINE_DELETED**: the number of rows deleted. This is an After variable and it returns an integer.
- **NB_LINE_REJECTED**: the number of rows rejected. This is an After variable and it returns an integer.
- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

- **Usage rule**: This component offers the flexibility benefit of the DB query and covers all of the SQL queries possible.

  This component must be used as an output component. It allows you to carry out actions on a table or on the data of a table in a Vertica database. It also allows you to create a reject flow using a Row > Rejects link to filter data in error. For an example of *tMySqlOutput* in use, see Scenario: Retrieving data in error with a Reject link on page 2529.

Limitation

- **Limitation**: Due to license incompatibility, one or more JARs required to use this component are not provided.

  You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Related scenario

For *tVectorWiseOutput* related topics, see:
• **Scenario: Inserting a column and altering data using tMysqlOutput** on page 2521.
tVectorWiseRollback

Cancels transactions committed to the VectorWise database.

**tVectorWiseRollback Standard properties**

These properties are used to configure tVectorWiseRollback running in the Standard Job framework.

The Standard tVectorWiseRollback component belongs to the Databases family.

The component in this framework is generally available.

![Note:](image)

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component list</td>
<td>Select the <strong>tVectorWiseConnection</strong> component from the list if more than one connection is planned for the current job.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect log data at the component level. |

**Usage**

| Usage rule | This component is more commonly used with other tVectorWise* components, especially with the **tVectorWiseConnection** and **tVectorWiseCommit** components. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job.  

Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.  

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using **
Related scenario

For a `tVectorWiseRollback` related scenario, see *Scenario: Rollback from inserting data in mother/daughter tables* on page 2480.
tVectorWiseRow

Acts on the actual DB structure or on the data (although without handling data).

tVectorWiseRow is the specific component for this database query. It executes the SQL query stated in the specified database. The row suffix means the component implements a flow in the job design although it doesn’t provide output.

The SQLBuilder tool helps you write your SQL statements easily.

**tVectorWiseRow Standard properties**

These properties are used to configure tVectorWiseRow running in the Standard Job framework.

The Standard tVectorWiseRow component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property type</td>
<td>Either Built-in or Repository.</td>
</tr>
<tr>
<td>Built-in</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td>Repository</td>
<td>Select the repository file where the properties are stored. The fields that follow after are completed automatically using the data retrieved.</td>
</tr>
</tbody>
</table>

**Use an existing connection**

Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.

**Note:**

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see [Talend Studio User Guide](#).

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>Database server IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>Listening port number of the DB server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Name of the database.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>DB user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td><strong>Schema</strong> and <strong>Edit Schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td><strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <a href="#">Talend Studio User Guide</a>.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <a href="#">Talend Studio User Guide</a>.</td>
<td></td>
</tr>
<tr>
<td><strong>Table Name</strong></td>
<td>Name of the table to be processed.</td>
</tr>
<tr>
<td><strong>Query type</strong></td>
<td>Either <strong>Built-in</strong> or <strong>Repository</strong>.</td>
</tr>
<tr>
<td><strong>Built-in</strong>: Fill in the query statement manually or build it graphically using the SQLBuilder.</td>
<td></td>
</tr>
<tr>
<td><strong>Repository</strong>: Select the relevant query stored in the Repository. The <strong>Query</strong> field is filled in accordingly.</td>
<td></td>
</tr>
<tr>
<td><strong>Guess Query</strong></td>
<td>Click this button to generate a query that corresponds to your table schema in the <strong>Query</strong> field.</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>Enter your DB query taking care to sequence the fields properly in order to match the schema definition.</td>
</tr>
<tr>
<td><strong>Die on error</strong></td>
<td>This check box is selected by default. Clear the check box to skip the row on error and complete the process for error-free rows. If needed, you can retrieve the rows on error via a <strong>Row &gt; Rejects</strong> link.</td>
</tr>
</tbody>
</table>

### Advanced settings

| **Propagate QUERY’s recordset** | Select this check box to insert the result of the query into a **COLUMN** of the current flow. Select this column from the **use column** list.  

**Note:**   
This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the QUERY’s recordset should be set to the type of **Object** and this component is usually followed by **tParseRecordSet**. |
|-------------------|--------------------------------------------------------------------------------------------------|
| **Use PreparedStatement** | Select this check box if you want to query the database using a PreparedStatement. In the **Set PreparedStatement Parameter** table, define the parameters represented by “?” in the SQL instruction of the **Query** field in the **Basic Settings** tab.  

**Parameter Index:** Enter the parameter position in the SQL instruction.  

**Parameter Type:** Enter the parameter type.  

**Parameter Value:** Enter the parameter value.  

**Note:**  
This option is very useful if you need to execute the same query several times. Performance levels are increased |
| **Commit every** | Number of rows to be completed before committing batches of rows together into the DB. This option ensures transaction quality (but not rollback) and above all better performance on executions. |
| **tStat Catcher Statistics** | Select this check box to collect log data at the component level. |

### Global Variables

| **Global Variables** | **QUERY:** the query statement being processed. This is a Flow variable and it returns a string.  

**ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable. |
|---------------------|--------------------------------------------------------------------------------------------------|
variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component offers the flexibility of the DB query and covers all possible SQL queries.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the <strong>Install</strong> button on the <strong>Component</strong> tab view. You can also find out and add all missing JARs easily on the <strong>Modules</strong> tab in the <strong>Integration</strong> perspective of your studio. You can find more details about how to install external modules in Talend Help Center (<a href="https://help.talend.com">https://help.talend.com</a>).</td>
</tr>
</tbody>
</table>

### Related scenario

For related topics, see:

- **Scenario: Combining two flows for selective output** on page 2558
- **Procedure** on page 625.
- **Scenario: Removing and regenerating a MySQL table index** on page 2552.
Load data into a Vertica database table from a local file using the Vertica COPY SQL statement.

For more information about the Vertica COPY SQL statement, see COPY.

The tVerticaOutputBulk component and the tVerticaBulkExec component are generally used together as parts of a two-step process. In the first step, an output file is generated. In the second step, the file is used in a bulk load operation to feed a database. These two steps are fused together in the tVerticaOutputBulkExec component. The advantage of using two separate components is that the data can be transformed before it is loaded into the database.

**tVerticaBulkExec Standard properties**

These properties are used to configure tVerticaBulkExec running in the Standard Job framework.

The Standard tVerticaBulkExec component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click Apply.</td>
</tr>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td><strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td><strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the version of the database.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined. When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
</tbody>
</table>
1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Host</th>
<th>The IP address or hostname of the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>The listening port number of the database.</td>
</tr>
<tr>
<td>Database</td>
<td>The name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>The schema of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>The database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>The name of the table into which data will be written.</td>
</tr>
</tbody>
</table>

**Action on table**
- **Default**: No operation is carried out.
- **Drop and create table**: The table is removed and created again.
- **Create table**: The table does not exist and gets created.
- **Create table if does not exist**: The table is created if it does not exist.
- **Drop table if exist and create**: The table is removed if it already exists and created again.
- **Clear table**: The table content is deleted. You have the possibility to rollback the operation.

**Schema and Edit schema**
- **Built-In**: You create and store the schema locally for this component only. Related topic: see Talend Studio User Guide.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see Talend Studio User Guide.
When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th>Use schema columns for Copy</th>
<th>Select this check box to use the column option in the COPY statement so that you can restrict the load to one or more specified columns in the table. For more information, see the Vertica COPY SQL Statement.</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>The path to the file from which data will be loaded. The file should be located on the same machine where the Studio is installed or where the Job using this component is deployed. This property is available only when there is no input flow.</td>
</tr>
<tr>
<td>Compression mode</td>
<td>Select the compression mode for the file from which data will be loaded. This property is available only when you are using Vertica 6.0 and later.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Additional JDBC Parameters</th>
<th>Specify additional JDBC parameters for the database connection being created. This property is not available when the Use an existing connection check box is selected.</th>
</tr>
</thead>
</table>
| Action on data            | Select an action that will be performed on the data of the table defined.  
  - **Bulk insert**: Insert multiple rows into the table at once instead of doing single row inserts. If duplicates are found, the Job stops. |
- **Bulk update**: Make simultaneous updates to multiple rows.

<table>
<thead>
<tr>
<th><strong>Stream name</strong></th>
<th>The stream name of a load, which helps identify a particular load. This property is available only when you are using Vertica 6.0 and later.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Write to ROS (Read Optimized Store)</strong></td>
<td>Select this check box to store data in a physical storage area, in order to optimize the reading, as the data is compressed and pre-sorted.</td>
</tr>
<tr>
<td><strong>Exit Job on no rows loaded</strong></td>
<td>The Job automatically stops if no row has been loaded.</td>
</tr>
<tr>
<td><strong>Missing columns as null</strong></td>
<td>Select this check box to insert NULL values for the missing columns when there is insufficient data to match the columns specified in the schema. This property is available only when you are using Vertica 6.0 and later.</td>
</tr>
<tr>
<td><strong>Skip Header</strong></td>
<td>Select this check box and in the field displayed next to it, specify the number of records to skip in the file. This property is available only when you are using Vertica 6.0 and later.</td>
</tr>
<tr>
<td><strong>Record terminator</strong></td>
<td>Select this check box and in the field displayed next to it, specify the literal character string used to indicate the end of each record in the file. This property is available only when you are using Vertica 6.0 and later.</td>
</tr>
<tr>
<td><strong>Enclosed by character</strong></td>
<td>Select this check box to set the character within which data is enclosed. This property is available only when you are using Vertica 6.0 and later.</td>
</tr>
<tr>
<td><strong>Escape char</strong></td>
<td>Select this check box and in the field displayed specify the character to be escaped when loading data into Vertica. By default, the check box is selected and the default escape character is .</td>
</tr>
<tr>
<td><strong>Fields terminated by</strong></td>
<td>The character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td><strong>Null String</strong></td>
<td>The string displayed to indicate that the value is null.</td>
</tr>
<tr>
<td><strong>Reject not fitted values</strong></td>
<td>Select this check box to reject data rows of type char, varchar, binary, and varbinary if they do not fit the target table. This property is available only when you are using Vertica 6.0 and later.</td>
</tr>
<tr>
<td><strong>Maximum number of rejected records</strong></td>
<td>Select this check box and in the field displayed next to it, specify the maximum number of records that can be rejected before a load fails.</td>
</tr>
</tbody>
</table>
This property is available only when you are using Vertica 6.0 and later.

**Stop and rollback if any row is rejected**
Select this check box to stop and roll back a load without loading any data if any row is rejected.
This property is available only when you are using Vertica 6.0 and later.

**Don't commit**
Select this check box to perform a bulk load transaction without committing the results automatically. This is useful if you want to execute multiple bulk loads in a single transaction.
This property is available only when you are using Vertica 6.0 and later.

**Rejected data file**
Specify the file into which rejected rows will be written.
This property is available only when Bulk insert is selected from the Action on data drop-down list.

**Exception log file**
Specify the file into which the exception log will be written. This log explains why each rejected row was rejected.
This property is available only when Bulk insert is selected from the Action on data drop-down list.

**tStatCatcher Statistics**
Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEPTED_ROW_NUMBER</td>
<td>The number of rows loaded into the database. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>REJECTED_ROW_NUMBER</td>
<td>The number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

**Usage**

**Usage rule**
Talend Studio and the Vertica database create very fast and affordable data warehouse and data mart applications. For more information about how to configure Talend Studio to connect to Vertica, see Talend and HP Vertica Tips and Techniques.
You can use this component in either of the following two ways to write data into Vertica.
- It can be used as a Standalone component of a Subjob to write data into Vertica from a file generated by a tVerticaOutputBulk component.
You can link a tFileInputRaw component to it via a Row > Main connection to feed data into Vertica. In this way, the tFileInputRaw component should be in the Stream the file mode and there should be only one column of Object type defined in its schema.

**Dynamic settings**

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

**Related scenarios**

For related topics, see:

- Scenario: Inserting transformed data in MySQL database on page 2537.
- Scenario: Inserting data in bulk in MySQL database on page 2544.
- Scenario: Truncating and inserting file data into an Oracle database on page 2752.
### tVerticaClose

Closes an active connection to a Vertica database.

### tVerticaClose Standard properties

These properties are used to configure tVerticaClose running in the Standard Job framework.

The Standard tVerticaClose component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

#### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the component in which the relevant reused connection is configured.</td>
</tr>
</tbody>
</table>

#### Advanced settings

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

#### Usage

**Usage rule**

This component is more commonly used with other tVertica* components, especially with the tVerticaConnection and tVerticaCommit components.

Talend Studio and the Vertica database create very fast and affordable data warehouse and data mart applications. For more information about how to configure Talend Studio to connect to Vertica, see Talend and HP Vertica Tips and Techniques.

**Dynamic settings**

Click the [+] button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.
Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenarios

No scenario is available for the Standard version of this component yet.
tVerticaCommit

Commits in one go a global transaction using a unique connection instead of doing that on every row or every batch and thus provides gain in performance.

### tVerticaCommit Standard properties

These properties are used to configure tVerticaCommit running in the Standard Job framework.

The Standard tVerticaCommit component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the component in which the relevant reused connection is configured.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Select this check box to close the database connection once the component has performed its task. Clear this check box to continue to use the selected connection once the component has performed its task. If this component is linked to your Job via a Row &gt; Main connection, your data will be committed row by row. In this case, do not select the Close connection check box or your connection will be closed before the end of the first row commit.</td>
</tr>
</tbody>
</table>

### Advanced settings

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

### Usage

**Usage rule**

This component is more commonly used with other tVertica* components, especially with the tVerticaConnection and tVerticaRollback components.

Talend Studio and the Vertica database create very fast and affordable data warehouse and data mart applications. For more information about how to
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |

**Related scenario**

For tVerticaCommit related scenario, see Scenario: Inserting data in mother/daughter tables on page 2476
tVerticaConnection

Opens a connection to the specified database that can then be reused in the subsequent subjob or subjobs.

**tVerticaConnection Standard properties**

These properties are used to configure tVerticaConnection running in the Standard Job framework. The Standard tVerticaConnection component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see *Dynamic database components* on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Type</strong></td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td><strong>DB Version</strong></td>
<td>Select the version of the database.</td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td>The IP address or hostname of the database.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>The listening port number of the database.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>The name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>The schema of the database.</td>
</tr>
<tr>
<td><strong>Username</strong> and <strong>Password</strong></td>
<td>The database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
</tbody>
</table>
**Additional JDBC Parameters**

Specify additional JDBC parameters for the database connection being created.

**Use or register a shared DB Connection**

Select this check box to share your database connection or fetch a database connection shared by a parent or child Job, and in the **Shared DB Connection Name** field displayed, enter the name for the shared database connection. This allows you to share one single database connection (except the database schema setting) among several database connection components from different Job levels that can be either parent or child.

This option is incompatible with the **Use dynamic job** and **Use an independent process to run subjob** options of the **tRunJob** component. Using a shared connection together with a **tRunJob** component with either of these two options enabled will cause your Job to fail.

**Advanced settings**

**Auto Commit**

Select this check box to commit any changes to the database automatically upon the transaction.

With this check box selected, you cannot use the corresponding commit component to commit changes to the database; likewise, when using the corresponding commit component, this check box has to be cleared. By default, the auto commit function is disabled and changes must be committed explicitly using the corresponding commit component.

Note that the auto commit function commits each SQL statement as a single transaction immediately after the statement is executed while the commit component does not commit only until all of the statements are executed. For this reason, if you need more room to manage your transactions in a Job, it is recommended to use the commit component.

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Usage**

**Usage rule**

This component is more commonly used with other tVertica components, especially with the **tVerticaCommit** and **tVerticaRollback** components.

Talend Studio and the Vertica database create very fast and affordable data warehouse and data mart applications. For more information about how to configure **Talend Studio** to connect to Vertica, see **Talend and HP Vertica Tips and Techniques**.
Related scenario

For `tVerticaConnection` related scenario, see `tMysqlConnection` on page 2475.
tVerticaInput

Retrieves data from a Vertica database table based on a SQL query.

**tVerticaInput Standard properties**

These properties are used to configure tVerticaInput running in the Standard Job framework.

The Standard tVerticaInput component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
</table>
| Property Type | Select the way the connection details will be set.  
| | • **Built-In**: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.  
| | • **Repository**: The connection details stored centrally in **Repository > Metadata** will be reused by this component. You need to click the [...] button next to it and in the pop-up **Repository Content** dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in. |
| ![icon] | Click the icon to open a database connection wizard and store the database connection parameters you set in the component.  
| | For more information about setting up and storing database connection parameters, see Talend Studio User Guide. |
| DB Version | Select the version of the database. |
| Use an existing connection | Select this check box and in the **Component List** click the relevant connection component to reuse the connection details you already defined.  
| | When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:  
| | 1. In the parent level, register the database connection to be shared in the **Basic settings** view of the |
connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Host</th>
<th>The IP address or hostname of the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>The listening port number of the database.</td>
</tr>
<tr>
<td>Database</td>
<td>The name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>The schema of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>The database user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
<tr>
<td>Table Name</td>
<td>The name of the table from which data will be retrieved.</td>
</tr>
<tr>
<td>Query Type and Query</td>
<td>Specify the database query statement paying particularly attention to the properly sequence of the fields which must correspond to the schema definition.</td>
</tr>
</tbody>
</table>
**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Additional JDBC Parameters</strong></th>
<th>Specify additional JDBC parameters for the database connection being created. This property is not available when the Use an existing connection check box is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trim all the String/Char columns</strong></td>
<td>Select this check box to remove leading whitespace and trailing whitespace from all String/Char columns.</td>
</tr>
<tr>
<td><strong>Trim column</strong></td>
<td>Select the check box for corresponding column to remove leading whitespace and trailing whitespace from it. This property is not available when the Trim all the String/Char columns check box is selected.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

**Global Variables**

| **NB_LINE** | The number of rows processed. This is an After variable and it returns an integer. |
| **QUERY** | The query statement being processed. This is a Flow variable and it returns a string. |
| **ERROR_MESSAGE** | The error message generated by the component when an error occurs. This is an After variable and it returns a string. |

**Usage**

**Usage rule**

This component covers all possible SQL queries for the Vertica database.

Talend Studio and the Vertica database create very fast and affordable data warehouse and data mart applications. For more information about how to configure Talend Studio to connect to Vertica, see Talend and HP Vertica Tips and Techniques.

**Dynamic settings**

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose...
your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

### Related scenarios

For related scenarios, see:

- Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489.
tVerticaOutput

Inserts, updates, deletes, or copies data from an incoming flow into a Vertica database table.

**tVerticaOutput Standard properties**

These properties are used to configure tVerticaOutput running in the Standard Job framework.

The Standard tVerticaOutput component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

### Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In:</strong> The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository:</strong> The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td><img src="image" alt="icon" /></td>
<td>Click the icon to open a database connection wizard and store the database connection parameters you set in the component.</td>
</tr>
<tr>
<td></td>
<td>For more information about setting up and storing database connection parameters, see Talend Studio User Guide.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the version of the database.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the Basic settings view of the</td>
</tr>
</tbody>
</table>
connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Host</th>
<th>The IP address or hostname of the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>The listening port number of the database.</td>
</tr>
<tr>
<td>Database</td>
<td>The name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>The schema of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>The database user authentication data.</td>
</tr>
<tr>
<td></td>
<td>To enter the password, click the [..] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Table</td>
<td>The name of the table into which data will be written.</td>
</tr>
<tr>
<td>Action on table</td>
<td>Select an operation to be performed on the table defined.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Default</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Drop and create table</strong>: The table is removed and created again.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Create table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Create table if does not exist</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Drop table if exist and create</strong>: The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Clear table</strong>: The table content is deleted. You have the possibility to rollback the operation.</td>
</tr>
<tr>
<td>Use &quot;drop cascade&quot;</td>
<td>Select this check box to remove all objects related to the table which will be dropped.</td>
</tr>
<tr>
<td></td>
<td>This property is available only when a table drop related option is selected from the <strong>Action on table</strong> list.</td>
</tr>
<tr>
<td>Action on data</td>
<td>Select an action to be performed on data of the table defined.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Insert</strong>: Add new entries to the table. If duplicates are found, job stops.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update</strong>: Make changes to existing entries.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Insert or update</strong>: Insert a new record. If the record with the given reference already exists, an update would be made.</td>
</tr>
</tbody>
</table>
- **Update or insert**: Update the record with the given reference. If the record does not exist, a new record would be inserted.

- **Delete**: Remove entries corresponding to the input flow.

- **Copy**: Read data from a text file and insert tuples of entries into the WOS (Write Optimized Store) or directly into the ROS (Read Optimized Store). This option is ideal for bulk loading. For further information, see Vertica SQL Reference Manual.

It is necessary to specify at least one column as a primary key on which the **Update** and **Delete** operations are based. You can do that by clicking **Edit Schema** and selecting the check box(es) next to the column(s) you want to set as primary key(s). For an advanced use, click the **Advanced settings** view where you can simultaneously define primary keys for the **Update** and **Delete** operations. To do that, select the **Use field options** check box and then in the **Update Key** column, select the check boxes next to the column names you want to use as a base for the **Update** operation. Do the same in the **Deletion key** column for the **Delete** operation.

**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word **line** when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see **Talend Studio User Guide**.

- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see **Talend Studio User Guide**.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.

- **View schema**: choose this option to view the schema only.

- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No**.
upon completion and choose this schema metadata again in the [Repository Content] window.

<table>
<thead>
<tr>
<th><strong>Die on error</strong></th>
<th>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a Row &gt; Reject connection.</th>
</tr>
</thead>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th><strong>Additional JDBC Parameters</strong></th>
<th>Specify additional JDBC parameters for the database connection being created. This property is not available when the Use an existing connection check box is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abort on error</strong></td>
<td>Select this check box to stop the Copy operation if any row is rejected and roll back the operation. Thus no data is loaded. This property is available only when COPY is selected from the Action on data drop-down list.</td>
</tr>
<tr>
<td><strong>Maximum rejects</strong></td>
<td>Type in a number to set the REJECTMAX command used by Vertica, which indicates the upper limit on the number of logical records to be rejected before a load fails. If not specified or if value is 0, an unlimited number of rejections are allowed. This property is available only when COPY is selected from the Action on data drop-down list.</td>
</tr>
<tr>
<td><strong>No commit</strong></td>
<td>Select this check box to prevent the current transaction from committing automatically. This property is available only when COPY is selected from the Action on data drop-down list.</td>
</tr>
<tr>
<td><strong>Exception file</strong></td>
<td>Type in the path to, or browse to the file in which messages are written indicating the input line number and the reason for each rejected data record. This property is available only when COPY is selected from the Action on data drop-down list.</td>
</tr>
<tr>
<td><strong>Exception file node</strong></td>
<td>Type in the node of the exception file. If not specified, operations default to the query's initiator node. This property is available only when COPY is selected from the Action on data drop-down list.</td>
</tr>
<tr>
<td><strong>Rejected data file</strong></td>
<td>Type in the path to, or browse to the file in which to write rejected rows. This file can then be edited to resolve problems and reloaded. This property is available only when COPY is selected from the Action on data drop-down list.</td>
</tr>
</tbody>
</table>
| **Rejected data file node** | Type in the node of the rejected data file. If not specified, operations default to the query’s initiator node.  
This property is available only when **COPY** is selected from the **Action on data** drop-down list. |
| **Commit every** | Specify the number of rows to be processed before committing batches of rows together into the database.  
This option ensures transaction quality (but not rollback) and, above all, better performance at executions. |
| **Use batch mode** | Select this check box to activate the batch mode for data processing, and in the **Batch size** field displayed, specify the number of records to be processed in each batch.  
This property is available only when **Insert**, **Update**, **Delete** or **Copy** is selected from the **Action on data** drop-down list. |
| **Additional Columns** | This option allows you to call SQL functions to perform actions on columns, which are not insert, nor update or delete actions, or action that require particular preprocessing. It is not offered if you create (with or without drop) a database table.  
- **Name**: Type in the name of the schema column to be altered or inserted as new column.  
- **DataType**: Type in the data type for the new column.  
- **SQL expression**: Type in the SQL statement to be executed in order to alter or insert the relevant column data.  
- **Position**: Select **Before**, **Replace** or **After** following the action to be performed on the reference column.  
- **Reference column**: Select a column of reference that the component can use to place or replace the new or altered column. |
| **Use field options** | Select the check box for the corresponding column to customize a request, particularly if multiple actions are being carried out on the data.  
- **Update Key**: Select the check box for the corresponding column based on which the data is updated.  
- **Deletion Key**: Select the check box for the corresponding column based on which the data is deleted.  
- **Updatable**: Select the check box if the data in the corresponding column can be updated.  
- **Insertable**: Select the check box if the data in the corresponding column can be inserted. |
| **Enable debug mode** | Select this check box to display each step during processing entries in a database. |
### Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE</td>
<td>The number of rows processed. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_COPIED</td>
<td>The number of rows copied. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_DELETED</td>
<td>The number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_INSERTED</td>
<td>The number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_REJECTED</td>
<td>The number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_UPDATED</td>
<td>The number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

### Usage

#### Usage rule

This component is usually used as an output component. It allows you to carry out actions on a table or on data of a table in a Vertica database. It also allows you to create a reject flow using a **Row > Rejects** link to filter data in error. For an example of **tMysqlOutput** in use, see **Scenario: Retrieving data in error with a Reject link** on page 2529.

Talend Studio and the Vertica database create very fast and affordable data warehouse and data mart applications. For more information about how to configure **Talend Studio** to connect to Vertica, see **Talend and HP Vertica Tips and Techniques**.

#### Dynamic settings

Click the `[+]` button to add a row in the table and fill the **Code** field with a context variable to choose your database connection dynamically from multiple...
connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The **Dynamic settings** table is available only when the **Use an existing connection** check box is selected in the **Basic settings** view. Once a dynamic parameter is defined, the **Component List** box in the **Basic settings** view becomes unusable.

For examples on using dynamic parameters, see **Scenario: Reading data from databases through context-based dynamic connections** on page 2497 and **Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters** on page 489.

For more information on **Dynamic settings** and context variables, see Talend Studio User Guide.

---

**Related scenarios**

For **tVerticaOutput** related topics, see:

- **Scenario: Inserting a column and altering data using tMysqlOutput** on page 2521.
tVerticaOutputBulk

Prepares a file to be used by the tVerticaBulkExec component to feed a Vertica database.

The tVerticaOutputBulk component and the tVerticaBulkExec component are generally used together as parts of a two step process. In the first step, an output file is generated. In the second step, the file is used in a bulk load operation to feed a database. These two steps are fused together in the tVerticaOutputBulkExec component. The advantage of using two separate components is that the data can be transformed before it is loaded into the database.

tVerticaOutputBulk Standard properties

These properties are used to configure tVerticaOutputBulk running in the Standard Job framework.

The Standard tVerticaOutputBulk component belongs to the Databases family.

The component in this framework is generally available.

Note:
This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the file path and the schema will be set.</td>
</tr>
<tr>
<td></td>
<td>• Built-In: The file path and the schema will be set locally for this component.</td>
</tr>
<tr>
<td></td>
<td>• Repository: The file details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the file to be reused, and all related properties will be automatically filled in.</td>
</tr>
<tr>
<td>File Name</td>
<td>The path to the file to be generated.</td>
</tr>
<tr>
<td></td>
<td>This file is generated on the same machine where your Studio is installed or where your Job using this component is deployed.</td>
</tr>
<tr>
<td>Append</td>
<td>Select this check box to add new rows at the end of the file.</td>
</tr>
<tr>
<td>Schema and Edit schema</td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.</td>
</tr>
</tbody>
</table>
- **Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center ([https://help.talend.com](https://help.talend.com)).

Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.

- **View schema**: choose this option to view the schema only.

- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **[Repository Content]** window.

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Row Separator</strong></th>
<th>The separator used to identify the end of a row.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Include Header</strong></td>
<td>Select this check box to include the column header to the file.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select an encoding method from the list, or select <strong>CUSTOM</strong> and define it manually. This field is compulsory for database data handling.</td>
</tr>
<tr>
<td><strong>tStatCatcher Statistics</strong></td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th><strong>NB_LINE</strong></th>
<th>The number of rows processed. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ERROR_MESSAGE</strong></td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>
### Usage

| Usage rule | This component is more commonly used with the **tVerticaBulkExec** component. Used together, they offer gains in performance while feeding a Vertica database. Talend Studio and the Vertica database create very fast and affordable data warehouse and data mart applications. For more information about how to configure *Talend Studio* to connect to Vertica, see *Talend and HP Vertica Tips and Techniques*. |

### Related scenarios

For use cases in relation with **tVerticaOutputBulk**, see the following scenarios:

- **Scenario: Inserting transformed data in MySQL database** on page 2537.
- **Scenario: Inserting data in bulk in MySQL database** on page 2544.
tVerticaOutputBulkExec

Receives data from a preceding component, writes data into a local file, and loads data into a Vertica database from the file using the Vertica COPY SQL statement.

For more information about the Vertica COPY SQL statement, see COPY.

The tVerticaOutputBulk component and the tVerticaBulkExec component are generally used together as parts of a two step process. In the first step, an output file is generated. In the second step, the file is used in a bulk load operation to feed a database. These two steps are fused together in the tVerticaOutputBulkExec component. The advantage of using two separate components is that the data can be transformed before it is loaded into the database.

tVerticaOutputBulkExec Standard properties

These properties are used to configure tVerticaOutputBulkExec running in the Standard Job framework.

The Standard tVerticaOutputBulkExec component belongs to the Databases family.

The component in this framework is generally available.

Note:

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

Basic settings

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Select a type of database from the list and click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>- <em>Built-In</em>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>- <em>Repository</em>: The connection details stored centrally in <strong>Repository &gt; Metadata</strong> will be reused by this component. You need to click the [...] button next to it and in the pop-up <strong>Repository Content</strong> dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the version of the database.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the
two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the **Basic settings** view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.

For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th>Host</th>
<th>The IP address or hostname of the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>The listening port number of the database.</td>
</tr>
<tr>
<td>DB Name</td>
<td>The name of the database.</td>
</tr>
<tr>
<td>Schema</td>
<td>The schema of the database.</td>
</tr>
<tr>
<td>Username and Password</td>
<td>The database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Action on data</td>
<td>Select an action that will be performed on the data of the table defined.</td>
</tr>
<tr>
<td>• Bulk insert</td>
<td>Insert multiple rows into the table at once instead of doing single row inserts. If duplicates are found, the Job stops.</td>
</tr>
<tr>
<td>• Bulk update</td>
<td>Make simultaneous updates to multiple rows.</td>
</tr>
<tr>
<td>Table</td>
<td>The name of the table into which data will be written.</td>
</tr>
<tr>
<td>Action on table</td>
<td>Select an operation to be performed on the table defined.</td>
</tr>
<tr>
<td>• Default</td>
<td>No operation is carried out.</td>
</tr>
<tr>
<td>• Drop and create table</td>
<td>The table is removed and created again.</td>
</tr>
<tr>
<td>• Create table</td>
<td>The table does not exist and gets created.</td>
</tr>
<tr>
<td>• Create table if does not exist</td>
<td>The table is created if it does not exist.</td>
</tr>
<tr>
<td>• Drop table if exist and create</td>
<td>The table is removed if it already exists and created again.</td>
</tr>
<tr>
<td>• Clear table</td>
<td>The table content is deleted. You have the possibility to rollback the operation.</td>
</tr>
</tbody>
</table>
| Schema and Edit schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the
next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

When the schema to be reused has default values that are integers or functions, ensure that these default values are not enclosed within quotation marks. If they are, you must remove the quotation marks manually.

You can find more details about how to verify default values in retrieved schema in Talend Help Center (https://help.talend.com).

Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.

- **View schema**: choose this option to view the schema only.

- **Change to built-in property**: choose this option to change the schema to *Built-in* for local changes.

- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the *Repository Content* window.

<table>
<thead>
<tr>
<th>File Name</th>
<th>The path to the file to be generated. This file is generated on the same machine where your Studio is installed or where your Job using this component is deployed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append</td>
<td>Select this check box to add new rows at the end of the file.</td>
</tr>
<tr>
<td>Use schema columns for Copy</td>
<td>Select this check box to use the column option in the COPY statement so that you can restrict the load to one or more specified columns in the table. For more information, see the Vertica COPY SQL Statement.</td>
</tr>
</tbody>
</table>

**Advanced settings**

<table>
<thead>
<tr>
<th>Additional JDBC Parameters</th>
<th>Specify additional JDBC parameters for the database connection being created. This property is not available when the <strong>Use an existing connection</strong> check box is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream name</td>
<td>The stream name of a load, which helps identify a particular load.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Write to ROS (Read Optimized Store)</strong></td>
<td>Select this check box to store data in a physical storage area, in order to optimize the reading, as the data is compressed and pre-sorted.</td>
</tr>
<tr>
<td><strong>Exit Job if no row was loaded</strong></td>
<td>The Job automatically stops if no row has been loaded.</td>
</tr>
<tr>
<td><strong>Missing columns as null</strong></td>
<td>Select this check box to insert NULL values for the missing columns when there is insufficient data to match the columns specified in the schema. This property is available only when you are using Vertica 6.0 and later.</td>
</tr>
<tr>
<td><strong>Skip Header</strong></td>
<td>Select this check box and in the field displayed next to it, specify the number of records to skip in the file. This property is available only when you are using Vertica 6.0 and later.</td>
</tr>
<tr>
<td><strong>Record terminator</strong></td>
<td>Select this check box and in the field displayed next to it, specify the literal character string used to indicate the end of each record in the file. This property is available only when you are using Vertica 6.0 and later.</td>
</tr>
<tr>
<td><strong>Enclosed by character</strong></td>
<td>Select this check box to set the character within which data is enclosed. This property is available only when you are using Vertica 6.0 and later.</td>
</tr>
<tr>
<td><strong>Field Separator</strong></td>
<td>The character, string or regular expression to separate fields.</td>
</tr>
<tr>
<td><strong>Null String</strong></td>
<td>The string displayed to indicate that the value is null.</td>
</tr>
<tr>
<td><strong>Include Header</strong></td>
<td>Select the check box to include the column header to the file.</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select an encoding method from the list, or select <strong>CUSTOM</strong> and define it manually. This field is compulsory for database data handling.</td>
</tr>
<tr>
<td><strong>Reject not fitted values</strong></td>
<td>Select this check box to reject data rows of type char, varchar, binary, and varbinary if they do not fit the target table. This property is available only when you are using Vertica 6.0 and later.</td>
</tr>
<tr>
<td><strong>Maximum number of rejected records</strong></td>
<td>Select this check box and in the field displayed next to it, specify the maximum number of records that can be rejected before a load fails. This property is available only when you are using Vertica 6.0 and later.</td>
</tr>
</tbody>
</table>
Stop and rollback if any row is rejected

Select this check box to stop and roll back a load without loading any data if any row is rejected.
This property is available only when you are using Vertica 6.0 and later.

Don't commit

Select this check box to perform a bulk load transaction without committing the results automatically. This is useful if you want to execute multiple bulk loads in a single transaction.
This property is available only when you are using Vertica 6.0 and later.

Rejected data file

Specify the file into which rejected rows will be written.
This property is available only when Bulk insert is selected from the Action on data drop-down list.

Exception log file

Specify the file into which the exception log will be written. This log explains why each rejected row was rejected.
This property is available only when Bulk insert is selected from the Action on data drop-down list.

tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEPTED_ROW_NUMBER</td>
<td>The number of rows loaded into the database. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>REJECTED_ROW_NUMBER</td>
<td>The number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

**Usage**

**Usage rule**

This component is mainly used when no particular transformation is required on the data to be loaded into the database.
Talend Studio and the Vertica database create very fast and affordable data warehouse and data mart applications. For more information about how to configure Talend Studio to connect to Vertica, see Talend and HP Vertica Tips and Techniques.

**Related scenarios**

For use cases in relation with **tVerticaOutputBulkExec**, see the following scenarios:
• **Scenario: Inserting transformed data in MySQL database** on page 2537.
• **Scenario: Inserting data in bulk in MySQL database** on page 2544.
tVerticaRollback

Cancels the transaction commit in the Vertica database.

**tVerticaRollback Standard properties**

These properties are used to configure tVerticaRollback running in the Standard Job framework.

The Standard tVerticaRollback component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component List</td>
<td>Select the component in which the relevant reused connection is configured.</td>
</tr>
<tr>
<td>Close Connection</td>
<td>Select this check box to close the database connection once the component has performed its task. Clear this check box to continue to use the selected connection once the component has performed its task.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at the Job level as well as at each component level. |

**Usage**

| Usage rule | This component is more commonly used with other tVertica* components, especially with the tVerticaConnection and tVerticaCommit components. Talend Studio and the Vertica database create very fast and affordable data warehouse and data mart applications. For more information about how to configure Talend Studio to connect to Vertica, see Talend and HP Vertica Tips and Techniques. |
| Dynamic settings | Click the [*] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful |
when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

### Related scenario

For **tVerticaRollback** related scenario, see Scenario: Rollback from inserting data in mother/daughter tables on page 2480.
tVerticaRow

Executes a Vertica SQL statement against a database table.

**tVerticaRow Standard properties**

These properties are used to configure tVerticaRow running in the Standard Job framework.

The Standard tVerticaRow component belongs to the Databases family.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click <strong>Apply</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in <strong>Repository &gt; Metadata</strong> will be reused by this component. You need to click the [...] button next to it and in the pop-up <strong>Repository Content</strong> dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the version of the database.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the <strong>Component List</strong> click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
<tr>
<td></td>
<td>When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:</td>
</tr>
<tr>
<td></td>
<td>1. In the parent level, register the database connection to be shared in the <strong>Basic settings</strong> view of the connection component which creates that very database connection.</td>
</tr>
<tr>
<td></td>
<td>2. In the child level, use a dedicated connection component to read that registered database connection.</td>
</tr>
</tbody>
</table>
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>The IP address or hostname of the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>The listening port number of the database.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>The name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>The schema of the database.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>The database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
</tbody>
</table>

**Schema and Edit schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

- **Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.
- **Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.

Click **Edit schema** to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

**Table Name**

The name of the table to be processed.

**Query Type and Query**

Specify the database query statement paying particularly attention to the properly sequence of the fields which must correspond to the schema definition.

- **Built-In**: Fill in the query statement in the Query field manually or click the [...] button next to the Query field to build the statement graphically using the SQLBuilder.
### tVerticaRow

- **Repository**: Select the relevant query stored in the Repository by clicking the [...] button next to it and in the pop-up Repository Content dialog box, select the query to be used, and the **Query** field will be automatically filled in.

### Die on error

Select the check box to stop the execution of the Job when an error occurs.

Clear the check box to skip any rows on error and complete the process for error-free rows.

When errors are skipped, you can collect the rows on error using a **Row > Reject** connection.

### Advanced settings

<table>
<thead>
<tr>
<th>Additional JDBC Parameters</th>
<th>Specify additional JDBC parameters for the database connection being created. This property is not available when the <strong>Use an existing connection</strong> check box is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propagate QUERY’s recordset</td>
<td>Select this check box to propagate the result of the query to the output flow. From the <strong>use column</strong> list displayed, you need to select a column into which the query result will be inserted. This option allows the component to have a different schema from that of the preceding component. Moreover, the column that holds the query’s recordset should be set to the Object type and this component is usually followed by a <strong>tParseRecordSet</strong> component.</td>
</tr>
<tr>
<td>Use PreparedStatement</td>
<td>Select this check box if you want to query the database using a prepared statement. In the <strong>Set PreparedStatement Parameters</strong> table displayed, specify the value for each parameter represented by a question mark ? in the SQL statement defined in the <strong>Query</strong> field.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Parameter Index</strong>: the position of the parameter in the SQL statement.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Parameter Type</strong>: the data type of the parameter.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Parameter Value</strong>: the value of the parameter. For a related use case of this property, see Scenario: Using PreparedStatement objects to query data on page 2553.</td>
</tr>
<tr>
<td>Commit every</td>
<td>Specify the number of rows to be processed before committing batches of rows together into the database. This option ensures transaction quality (but not rollback) and, above all, better performance at executions.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>
Global Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE_DELETED</td>
<td>The number of rows deleted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_INSERTED</td>
<td>The number of rows inserted. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_UPDATED</td>
<td>The number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>QUERY</td>
<td>The query statement being processed. This is a Flow variable and it returns a string.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>

Usage

Usage rule

This component offers the flexibility of the DB query and covers all possible SQL queries.

Talend Studio and the Vertica database create very fast and affordable data warehouse and data mart applications. For more information about how to configure Talend Studio to connect to Vertica, see Talend and HP Vertica Tips and Techniques.

Dynamic settings

Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio.

The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable.

For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide.

Related scenario

For related topics, see:
• **Scenario: Combining two flows for selective output** on page 2558
• **Procedure** on page 625.
• **Scenario: Removing and regenerating a MySQL table index** on page 2552.
tVerticaSCD

Tracks and reflects data changes in a dedicated Vertica SCD table.

**tVerticaSCD Standard properties**

These properties are used to configure tVerticaSCD running in the Standard Job framework.

The Standard tVerticaSCD component belongs to the Business Intelligence and the Databases families.

The component in this framework is generally available.

**Note:**

This component is a specific version of a dynamic database connector. The properties related to database settings vary depending on your database type selection.

For more information about dynamic database connectors, see Dynamic database components on page 598.

**Basic settings**

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a type of database from the list and click Apply.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Type</td>
<td>Select the way the connection details will be set.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: The connection details will be set locally for this component. You need to specify the values for all related connection properties manually.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: The connection details stored centrally in Repository &gt; Metadata will be reused by this component. You need to click the [...] button next to it and in the pop-up Repository Content dialog box, select the connection details to be reused, and all related connection properties will be automatically filled in.</td>
</tr>
<tr>
<td>DB Version</td>
<td>Select the version of the database.</td>
</tr>
<tr>
<td>Use an existing connection</td>
<td>Select this check box and in the Component List click the relevant connection component to reuse the connection details you already defined.</td>
</tr>
</tbody>
</table>

When a Job contains the parent Job and the child Job, if you need to share an existing connection between the two levels, for example, to share the connection created by the parent Job with the child Job, you have to:

1. In the parent level, register the database connection to be shared in the Basic settings view of the connection component which creates that very database connection.

2. In the child level, use a dedicated connection component to read that registered database connection.
For an example about how to share a database connection across Job levels, see *Talend Studio User Guide*.

<table>
<thead>
<tr>
<th><strong>Host</strong></th>
<th>The IP address or hostname of the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port</strong></td>
<td>The listening port number of the database.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>The name of the database.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>The schema of the database.</td>
</tr>
<tr>
<td><strong>Username and Password</strong></td>
<td>The database user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click <strong>OK</strong> to save the settings.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>The name of the table into which data will be written.</td>
</tr>
<tr>
<td><strong>Action on table</strong></td>
<td>Select an operation to be performed on the table defined.</td>
</tr>
<tr>
<td></td>
<td>• <strong>None</strong>: No operation is carried out.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Create table</strong>: The table does not exist and gets created.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Create table if does not exist</strong>: The table is created if it does not exist.</td>
</tr>
<tr>
<td><strong>Schema and Edit schema</strong></td>
<td>A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word <code>line</code> when naming the fields.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Built-In</strong>: You create and store the schema locally for this component only. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Repository</strong>: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see <em>Talend Studio User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td>Click <strong>Edit schema</strong> to make changes to the schema. Note that if you make changes, the schema automatically becomes built-in.</td>
</tr>
<tr>
<td></td>
<td>• <strong>View schema</strong>: choose this option to view the schema only.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Change to built-in property</strong>: choose this option to change the schema to <strong>Built-in</strong> for local changes.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Update repository connection</strong>: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select <strong>No</strong> upon completion and choose this schema metadata again in the [Repository Content] window.</td>
</tr>
</tbody>
</table>
Click the [...] button and in the **SCD component editor** dialog box displayed, configure the data flow for slowly changing dimension outputs.

For more information, see **SCD management methodology** on page 2566.

<table>
<thead>
<tr>
<th>Use memory saving mode</th>
<th>Select this check box to maximize system performance.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die on error</td>
<td>Select the check box to stop the execution of the Job when an error occurs. Clear the check box to skip any rows on error and complete the process for error-free rows. When errors are skipped, you can collect the rows on error using a <strong>Row &gt; Reject</strong> connection.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>Additional JDBC Parameters</th>
<th>Specify additional JDBC parameters for the database connection being created. This property is not available when the <strong>Use an existing connection</strong> check box is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>End date time details</td>
<td>Specify the time value of the SCD end date time setting in the format of <strong>HH:mm:ss</strong>. The default value is <strong>12:00:00</strong>. This property is available only when the SCD Type 2 method is used and <strong>Fixed year value</strong> is selected for creating the SCD Type 2 end date in the SCD editor.</td>
</tr>
<tr>
<td>Debug mode</td>
<td>Select this check box to display each step during processing entries in a database.</td>
</tr>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

<table>
<thead>
<tr>
<th>NB_LINE_INSERTED</th>
<th>The number of rows inserted. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB_LINE_UPDATED</td>
<td>The number of rows updated. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>NB_LINE_REJECTED</td>
<td>The number of rows rejected. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>The error message generated by the component when an error occurs. This is an After variable and it returns a string.</td>
</tr>
</tbody>
</table>
### Usage

| Usage rule | This component is usually used as an end component of a Job or Subjob and it always needs an input link. Talend Studio and the Vertica database create very fast and affordable data warehouse and data mart applications. For more information about how to configure Talend Studio to connect to Vertica, see Talend and HP Vertica Tips and Techniques. |
| Dynamic settings | Click the [+] button to add a row in the table and fill the Code field with a context variable to choose your database connection dynamically from multiple connections planned in your Job. This feature is useful when you need to access database tables having the same data structure but in different databases, especially when you are working in an environment where you cannot change your Job settings, for example, when your Job has to be deployed and executed independent of Talend Studio. The Dynamic settings table is available only when the Use an existing connection check box is selected in the Basic settings view. Once a dynamic parameter is defined, the Component List box in the Basic settings view becomes unusable. For examples on using dynamic parameters, see Scenario: Reading data from databases through context-based dynamic connections on page 2497 and Scenario: Reading data from different MySQL databases using dynamically loaded connection parameters on page 489. For more information on Dynamic settings and context variables, see Talend Studio User Guide. |
| Limitation | This component does not support using SCD type 0 together with other SCD types. |

### Related scenarios

For a similar scenario, see tMysqlSCD on page 2563.
tVtigerCRMInput

Extracts data from a module of a VtigerCRM database.

### tVtigerCRMInput Standard properties

These properties are used to configure tVtigerCRMInput running in the Standard Job framework.

The Standard tVtigerCRMInput component belongs to the Business family.

The component in this framework is generally available.

#### Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vtiger Version</td>
<td>Select the version of the Vtiger Web Services you want to use (either Vtiger 5.0 or Vtiger 5.1)</td>
</tr>
</tbody>
</table>

When the version to be used is Vtiger 5.0:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Address</td>
<td>Type in the IP address of the VtigerCRM server</td>
</tr>
<tr>
<td>Port</td>
<td>Type in the Port number to access the server</td>
</tr>
<tr>
<td>Vtiger Path</td>
<td>Type in the path to access the VtigerCRM server</td>
</tr>
<tr>
<td>Username and Password</td>
<td>Type in the user authentication data. To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings.</td>
</tr>
<tr>
<td>Version</td>
<td>Type in the version of VtigerCRM you are using.</td>
</tr>
<tr>
<td>Module</td>
<td>Select the relevant module in the list</td>
</tr>
<tr>
<td>Method</td>
<td>Select the relevant method in the list. The method specifies the action you can carry out on the VtigerCRM module selected.</td>
</tr>
</tbody>
</table>

### Schema and Edit Schema

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to Built-in for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the
Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.

In this component the schema is related to the Module selected.

When the version to be used to Vtiger 5.1:

<table>
<thead>
<tr>
<th><strong>Endpoint</strong></th>
<th>Type in the URL address of the invoked Web server.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Username</strong></td>
<td>Type in the user name to log in to the vTigerCRM.</td>
</tr>
<tr>
<td><strong>Access key</strong></td>
<td>Type in the access key for the user name.</td>
</tr>
<tr>
<td><strong>Query condition</strong></td>
<td>Type in the query to select the data to be extracted.</td>
</tr>
<tr>
<td><strong>Manual input of SQL query</strong></td>
<td>Manually type in your query in the corresponding field.</td>
</tr>
</tbody>
</table>

**Advanced settings**

| **tStatCatcher Statistics** | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

| **Global Variables** | **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer. **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box. A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it. For further information about variables, see **Talend Studio User Guide**. |

**Usage**

| **Usage rule** | Usually used as a Start component. An output component is required. |
Related scenarios

No scenario is available for the Standard version of this component yet.
tVtigerCRMOoutput

Writes data into a module of a VtigerCRM database.

**tVtigerCRMOoutput Standard properties**

These properties are used to configure tVtigerCRMOoutput running in the Standard Job framework.
The Standard tVtigerCRMOoutput component belongs to the Business family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vtiger Version</td>
<td>Select the version of the Vtiger Web Services you want to use (either Vtiger 5.0 or Vtiger 5.1)</td>
</tr>
<tr>
<td>Server Address</td>
<td>Type in the IP address of the VtigerCRM server.</td>
</tr>
<tr>
<td>Port</td>
<td>Type in the Port number to access the server.</td>
</tr>
<tr>
<td>Vtiger Path</td>
<td>Type in the path to access the server.</td>
</tr>
</tbody>
</table>
| Username and Password | Type in the user authentication data.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| Version           | Type in the version of VtigerCRM you are using. |
| Module            | Select the relevant module in the list |
| Method            | Select the relevant method in the list. The method specifies the action you can carry out on the VtigerCRM module selected. |
| Schema and Edit Schema | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields. 
Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
• View schema: choose this option to view the schema only.  
• Change to built-in property: choose this option to change the schema to Built-in for local changes.  
• Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the |
When the version to be used is Vtiger 5.1

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endpoint</td>
<td>Type in the URL address of the invoked Web server.</td>
</tr>
<tr>
<td>Username</td>
<td>Type in the user name to log in to the VtigerCRM.</td>
</tr>
<tr>
<td>Access key</td>
<td>Type in the access key for the user name.</td>
</tr>
<tr>
<td>Action</td>
<td>Insert or Update the data in the SugarCRM module.</td>
</tr>
<tr>
<td>Module</td>
<td>Select the relevant module in the list</td>
</tr>
</tbody>
</table>

**Schema and Edit Schema**

A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word line when naming the fields.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

In this component the schema is related to the Module selected.

**Die on error**

This check box is clear by default to skip the row on error and complete the process for error-free rows.

**Advanced settings**

**tStatCatcher Statistics**

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.
Global Variables

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>NB_LINE: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>

Usage

| Usage rule | Used as an output component. An Input component is required. |

Related scenarios

No scenario is available for the Standard version of this component yet.
tWaitForFile

Iterates on a directory and triggers the next component when the defined condition is met.

This component is used to put the component connected with it in waiting state. It then triggers that
component when the defined file operation occurs in the specified directory.

tWaitForFile Standard properties

These properties are used to configure tWaitForFile running in the Standard Job framework.
The Standard tWaitForFile component belongs to the Orchestration family.
The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (in seconds) between iterations</td>
<td>Set the time interval in seconds between each two iterations.</td>
</tr>
<tr>
<td>Max. number of iterations (infinite loop if empty)</td>
<td>Set the maximum number of iterations on the specified directory.</td>
</tr>
<tr>
<td>Directory to scan</td>
<td>Specify the directory to iterate on.</td>
</tr>
<tr>
<td>File mask</td>
<td>Type in the file mask to filter the files to be monitored. To monitor a given file, type in the file name.</td>
</tr>
<tr>
<td>Include subdirectories</td>
<td>Select this check box to iterate also on the subdirectories.</td>
</tr>
<tr>
<td>Case sensitive</td>
<td>Select this check box to activate case sensitivity.</td>
</tr>
<tr>
<td>Include present file</td>
<td>Select this check box to include the file in use.</td>
</tr>
<tr>
<td>Trigger action when</td>
<td>Select one of the following trigger conditions:</td>
</tr>
<tr>
<td></td>
<td>a file is created: triggers the next component when a file is created.</td>
</tr>
<tr>
<td></td>
<td>a file is deleted: triggers the next component when a file is removed.</td>
</tr>
<tr>
<td></td>
<td>a file is updated: triggers the next component when a file is modified.</td>
</tr>
<tr>
<td></td>
<td>a file is created or updated or deleted: triggers the next component when a file is created, or removed, or modified.</td>
</tr>
<tr>
<td>Then</td>
<td>Select what to do with the iteration loop after the next component is triggered:</td>
</tr>
<tr>
<td></td>
<td>continue loop: continues iterating until the number of iterations reaches the defined maximum value.</td>
</tr>
<tr>
<td></td>
<td>exit loop: stops iterating immediately.</td>
</tr>
</tbody>
</table>
A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema of this component is **Built-in** and cannot be updated.

**Advanced Settings**

**Wait for file to be released Every (in ms)**

Select this check box so that the next component is triggered only after the file operation is completed.

**Global Variables**

**Global Variables**

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

- **CURRENT_ITERATION**: the sequence number of the current iteration. This is a Flow variable and it returns an integer.

- **PRESENT_FILE**: the path and name of the present file. This is a Flow variable and it returns a string.

- **DELETED_FILE**: the path and name of the deleted file. This is a Flow variable and it returns a string.

- **CREATED_FILE**: the path and name of the created file. This is a Flow variable and it returns a string.

- **UPDATED_FILE**: the path and name of the updated file. This is a Flow variable and it returns a string.

- **FILENAME**: the name of the file being processed. This is a Flow variable and it returns a string.

- **NOT_UPDATED_FILE**: the path and name of the file that has not been updated. This is a Flow variable and it returns a string.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

**Usage**

**Usage rule**

This component plays the role of triggering the next component based on the defined condition. Therefore this component requires another component to be connected to it via a link.

**Connections**

Outgoing links (from this component to another):
Scenario 1: Waiting for a file to be created and stopping the iteration loop after a message is triggered

This scenario describes a Job that scans a directory and triggers a message showing file creation information on the console when a text file is created in that directory. Immediately after this, the scanning is ended, followed by another message showing the number of iterations finished.

Dropping and linking the components

Procedure
1. Drop three components from the Palette onto the design workspace: one tWaitForFile and two tJava components.
2. Connect tWaitForFile to the first tJava using a Row > Iterate connection.
3. Connect tWaitForFile to the second tJava using a Trigger > On Subjob Ok connection.

Configuring the components

Procedure
1. Double-click tWaitForFile to open its Basic settings view.
2. In the **Directory to scan** field, enter the directory to be scanned.

3. In the **File mask** field, enter "*.txt" to scan only text files.

4. From the **Trigger action when** drop-down list, select **a file is created** to trigger the first **tJava** when a text file is created.

5. From the **Then** drop-down list, select **exit loop** to stop the iteration loop immediately after the first **tJava** is triggered.

6. Double-click the first **tJava** to open its **Basic settings** view.

   ![tJava 1](image)

   In the **code** area, type in the following code:
   ```java
   System.out.println("A file was created at "+TalendDate.getCurrentDate());
   System.out.println("Name of the created file: "+(String)globalMap.get("tWaitForFile_1_CREATED_FILE"));
   ```

7. In the **code** area, type in the following code:
   ```java
   System.out.println("Iteration loop ended at " + TalendDate.getCurrentDate());
   System.out.println("Number of iterations finished: " + ((Integer)globalMap.get("tWaitForFile_1_CURRENT_ITERATION")));
   ```

8. Double-click the second **tJava** to open its **Basic settings** view.

9. In the **code** area, type in the following code:
   ```java
   System.out.println("\r\nIteration loop ended at " + TalendDate.getCurrentDate());
   System.out.println("Number of iterations finished: " + ((Integer)globalMap.get("tWaitForFile_1_CURRENT_ITERATION")));
   ```

**Saving and executing the Job**

**Procedure**

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to execute the Job.
3. Create a text file **exit.txt** in the defined directory.
When a text file is created, a message about file creation is displayed on the Run console. The iteration loop is ended immediately after that, followed by another message showing the time the iteration loop is ended and the number of iterations executed.

**Scenario 2: Waiting for a file to be created and continuing the iteration loop after a message is triggered**

Based on the previous scenario, this scenario describes a Job triggers a message on the console when a text file is created in the directory being scanned, without ending the scanning process immediately. The Job continues scanning the directory until the number of iterations reaches the defined maximum value.

**Procedure**

1. Double-click `tWaitForFile` to open its **Basic settings** view.

2. In the **Time (in seconds) between iterations** field, enter the time interval in seconds between each two iterations. In this example, it is set to 5.

3. In the **Max. number of iterations (infinite loop if empty)** field, enter the maximum number of iterations. In this example, it is set to 10.

4. From the **Then** drop-down list, select **continue loop** to continue the iteration loop after the message is triggered.

5. Press **Ctrl+S** to save the Job.

6. Press **F6** to execute the Job.
7. Create a text file `continue.txt` in the defined directory.

When a text file is created, a message about file creation is displayed on the Run console. The iteration loop is ended until the number of iterations reaches ten, followed by another message showing the time the iteration loop is ended and the number of iterations executed.
tWaitForSocket

Triggers a Job based on a defined condition.

tWaitForSocket component makes a loop on a defined port, to look for data, and triggers a subjob when the condition is met.

**tWaitForSocket Standard properties**

These properties are used to configure tWaitForSocket running in the Standard Job framework.
The Standard tWaitForSocket component belongs to the Orchestration family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Port</th>
<th>DB server listening port.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>End of line separator</strong></td>
<td>Enter the end of line separator to be used.</td>
</tr>
</tbody>
</table>
| **Then**      | Select the action to be carried out:  
|               | keep on listening  
|               | or  
|               | close socket |
| **Print client/server data** | Select this check box to display the client or server data. |

**Advanced settings**

| tStatCatcher Statistics | Select this check box to collect the log data at a component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  
|                 | **INPUT_DATA**: the data transmitted by the client. This is a Flow variable and it returns a string.  
|                 | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
|                 | To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
|                 | For further information about variables, see Talend Studio User Guide. |

4017
## Usage

### Usage rule

This is an input, trigger component for the subjob executed depending on the condition set. Hence, it needs to be connected to a subjob via an Iterate link.

### Connections

Outgoing links (from this component to another):

- **Row**: Iterate.
- **Trigger**: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error.

Incoming links (from one component to this one):

- **Row**: Iterate.
- **Trigger**: On Subjob Ok; On Subjob Error; Run if; On Component Ok; On Component Error; Synchronize; Parallelize.

For further information regarding connections, see Talend Studio User Guide.

## Related scenarios

No scenario is available for the Standard version of this component yet.
tWaitForSqlData

Iterates on a given connection for insertion or deletion of rows and triggers a subjob when a condition linked to SQL data presence is met.

**tWaitForSqlData Standard properties**

These properties are used to configure tWaitForSqlData running in the Standard Job framework.

The Standard tWaitForSqlData component belongs to the Orchestration family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait at each iteration (in seconds)</td>
<td>Set the time interval in seconds between each check for the sql data.</td>
</tr>
<tr>
<td>Max. iterations (infinite if empty)</td>
<td>Number of checks for sql data before the Jobs times out.</td>
</tr>
<tr>
<td>Use an existing connection/Component List</td>
<td>A connection needs to be open to allow the loop to check for sql data on the defined DB.</td>
</tr>
</tbody>
</table>

**Note:**

When a Job contains the parent Job and the child Job, Component list presents only the connection components in the same Job level, so if you need to use an existing connection from the other level, you can

From the available database connection component in the level where the current component is, select the **Use or register a shared DB connection** check box. For more information about this check box, see tMysqlConnection on page 2475 for the connection components according to the database you are using.

Otherwise, still in the level of the current component, deactivate the connection components and use **Dynamic settings** of the component to specify the intended connection manually. In this case, make sure the connection name is unique and distinctive all over through the two Job levels. For more information about **Dynamic settings**, see Talend Studio User Guide.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table to scan</td>
<td>Name of the table to be checked for insert or deletion</td>
</tr>
<tr>
<td>Trigger action when rowcount is</td>
<td>Select the condition to be met for the action to be carried out:</td>
</tr>
</tbody>
</table>

Equal to  Not Equal to  Greater than  Lower than  Greater or equal to  Lower or equal to
**Value**

- **Define the value to take into account.**

**Then**

- **Select the action to be carried out: either stop the iterations when the condition is met (exit loop) or continue the loop until the end of the max iteration number (continue loop).**

---

### Global Variables

**ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

**CURRENT_ITERATION:** the sequence number of the current iteration. This is a Flow variable and it returns an integer.

**ROW_COUNT:** the number of rows detected in the table. This is a Flow variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

---

### Usage

**Usage rule**

- Although this component requires a Connection component to open the DB access, it plays also the role of the start (or trigger) component of the subjob which gets executed under the condition described. Therefore this component requires a subjob to be connected to via an Iterate link.

---

### Scenario: Waiting for insertion of rows in a table

This scenario describes a Job reading a DB table and waiting for data to be put in this table in order for a subjob to be executed. When the condition of the data insertion in the table is met, then the subjob performs a Select* on the table and simply displays the content of the inserted data onto the standard console.
Drop the following components from the Palette onto the design workspace: tMySqlConnection, tWaitForSqlData, tMysqlInput, tLogRow.

Connect the tMySqlConnection component to the tWaitForSqlData using an OnSubjobOK link, available on the right-click menu.

Then connect the tWaitForSqlData component to the subjob using an Iterate link as no actual data is transferred in this part. Indeed, simply a loop is implemented by the tWaitForSqlData until the condition is met.

On the subjob to be executed if the condition is met, a tMysqlInput is connected to the standard console component, tLogRow. As the connection passes on data, use a Row main link.

Now, set the connection to the table to check at regular intervals. On the Basic Settings view of the tMysqlConnection Component tab, set the DB connection properties.

- Fill out the Host, Port, Database, Username, Password fields to open the connection to the Database table.
- Select the relevant Encoding if needed.
- Then select the tWaitForSqlData component, and on the Basic Setting view of the Component tab, set its properties.
- In the Wait at each iteration field, set the time in seconds you want to wait before the next iteration starts.
- In the Max iterations field, fill out the number of iterations max you want to have before the whole Job is forced to end.
- The tWaitForSqlData component requires a connection to be open in order to loop on the defined number of iteration. Select the relevant connection (if several) in the Component List combo box.
- In the Table to scan field, type in the name of the table in the DB to scan. In this example: test_datatypes.
• In the Trigger action when rowcount is and Value fields, select the condition to be met, for the subjob to be triggered. In this use case, the number of rows in the scanned table should be greater or equal to 1.

• In the Then field, select the action to be carried out when the condition is met before the number of iteration defined is reached. In this use case, as soon as the condition is met, the loop should be ended.

• Then set the subjob to be executed when the condition set is met. In this use case, the subjob simply selects the data from the scanned table and displays it on the console.

• Select the tMySqlInput component, and on the Basic Setting view of the Component tab, set the connection to the table.

  ![Connection Settings](image)

  - Use an existing connection check box selected.
  - Component list: tMysqlConnection_1.
  - Scheme: Repository.
  - Table Name: 'test_datatypes'.
  - Query Type: Built-In.
  - Query: `SELECT * FROM test_datatypes`.

• If the connection is set in the Repository, select the relevant entry on the list. Or alternatively, select the Use an existing connection check box and select the relevant connection component on the list.

• In this use case, the schema corresponding to the table structure is stored in the Repository.

• Fill out the Table Name field with the table the data is extracted from, Test_datatypes.

• Then in the Query field, type in the Select statement to extract the content from the table.

• No particular setting is required in the tLogRow component for this use case.

Then before executing the Job, make sure the table to scan (test_datatypes) is empty, in order for the condition (greater or equal to 1) to be met. Then execute the Job by pressing the F6 key on your keyboard. Before the end of the iterating loop, feed the test_datatypes table with one or more rows in order to meet the condition.

<table>
<thead>
<tr>
<th>id</th>
<th>log_msg</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>143.112.32.4 · · [04/Mar/2008 00:00:00 +0100] &quot;GET /c...</td>
</tr>
</tbody>
</table>
| 1  | 143.112.32.4 · · [04/Mar/2008 00:00:00 +0100] "GET /c...

The Job ends when this table insert is detected during the loop, and the table content is thus displayed on the console.
Starting job tWaitForSql at 16:55 06/03/2006.
2| 143.112.32.4 - - [04/Mar/2008 00:00:00 +0100] "GET /components/none HTTP/1.1" 404 354 "Mozilla/4.0 (compatible
1| 143.112.32.4 - - [04/Mar/2008 00:00:00 +0100] "GET /components/none HTTP/1.1" 404 354 "Mozilla/4.0 (compatible
Job tWaitForSql ended at 16:55 06/03/2006. [exit code=0]
tWarn

Triggers a warning often caught by the tLogCatcher component for exhaustive log.

Both tDie and tWarn components are closely related to the tLogCatcher component. They generally make sense when used alongside a tLogCatcher in order for the log data collected to be encapsulated and passed on to the output defined.

This component provides a priority-rated message to the next component. It does not stop your Job in case of error. If you want to kill a Job in case of error, see tDie on page 637.

**tWarn Standard properties**

These properties are used to configure tWarn running in the Standard Job framework.

The Standard tWarn component belongs to the Logs & Errors family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Warn message</th>
<th>Type in your warning message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>Define the code level.</td>
</tr>
<tr>
<td>Priority</td>
<td>Enter the priority level as an integer.</td>
</tr>
</tbody>
</table>

**Global Variables**

<table>
<thead>
<tr>
<th>Global Variables</th>
<th>WARN_MESSAGES: the warning message. This is an After variable and it returns a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WARN_CODE: the code level of the warning message. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>WARN_PRIORITY: the priority level of the warning message. This is an After variable and it returns an integer.</td>
</tr>
<tr>
<td></td>
<td>ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.</td>
</tr>
<tr>
<td></td>
<td>A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.</td>
</tr>
<tr>
<td></td>
<td>To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.</td>
</tr>
<tr>
<td></td>
<td>For further information about variables, see Talend Studio User Guide.</td>
</tr>
</tbody>
</table>
Usage

| Usage rule | Cannot be used as a start component. If an output component is connected to it, an input component should be preceding it. |

Related scenarios

For use cases in relation with tWarn, see tLogCatcher scenarios:

- Catching messages triggered by a tWarn component on page 1982
- Catching the message triggered by a tDie component on page 1984
tWebService

Calls a method via a Web service in order to retrieve the values of the parameters defined in the component editor.

⚠️ Warning:  
This component requires an Oracle JDK to be functional.

tWebService calls the defined method from the invoked Web service and returns the class as defined, based on the given parameters.

**tWebService Standard properties**

These properties are used to configure tWebService running in the Standard Job framework.

The Standard tWebService component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either Built-in or Repository.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Built-in: No property data stored centrally. Enter properties manually.</td>
</tr>
<tr>
<td></td>
<td>Repository: Select the repository file where properties are stored. The fields that come after are pre-filled in using the fetched data.</td>
</tr>
</tbody>
</table>

**Service configuration**

Click the three-dot button next to the field to open the component editor.

In this editor, you can:

- select the Web service you want to use,
- configure the input parameters of the Web service,
- configure the output parameters of the Web service.

These parameters will be used to retrieve and output specific data.

**Mapping links display as**

Auto: By default, the links between the input and output schemas and the Web service parameters are in the form of curves.

Curves: Links between the schema and the Web service parameters are in the form of curve.

Lines: Links between the schema and the Web service parameters are in the form of straight lines. This option slightly optimizes performance.

**Connection Time out**

Set a value in seconds for Web service connection timeout.

**Receive Time out**

Set a value in seconds for server answer.
**Input schema**

A schema is a row description. It defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository. This field is used to process the input schema. The option for this schema may be:

- **Built-in**: No property data stored centrally.
- **Repository**: Select the Repository file where properties are stored. The following fields are pre-filled in using fetched data.

**Edit Schema**

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Sync columns**

This button is available when an input link has been created. Click this button to retrieve the schema from the previous component connected in the Job.

**Output schema**

This field is used to process the output schema. The schema is either built-in or remote in the Repository and is configured the same way as the input schema is.

**Warning:**

*The input schema is not necessarily identical with the output schema.*

**Use NTLM**

Select this check box if you want to use the NTLM authentication protocol.

**Domain**: Name of the client domain,

**Host**: Client IP address.

**Need authentication**

Select this check box and enter a username and a password in the corresponding fields if this is necessary to access the service.

To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click **OK** to save the settings.

**Use http proxy**

Select this check box if you are using a proxy server and fill in the necessary information.
| Trust server with SSL | Select this check box to validate the server certificate to the client via an SSL protocol and fill in the corresponding fields:  
**TrustStore file:** enter the path (including filename) to the certificate TrustStore file that contains the list of certificates that the client trusts.  
**TrustStore password:** enter the password used to check the integrity of the TrustStore data. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Die on error</td>
<td>Clear this check box to skip the rows in error and to complete the process for the error free rows.</td>
</tr>
</tbody>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th>Temporary folder (for wsdl2java)</th>
<th>Set or browse to a temporary folder that you configured in order to store the wsdl files.</th>
</tr>
</thead>
<tbody>
<tr>
<td>tStatCatcher Statistics</td>
<td>Select this check box to gather the Job processing metadata at a Job level as well as at each component level.</td>
</tr>
</tbody>
</table>

### Global Variables

| Global Variables |  
**ERROR_MESSAGE:** the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.  
**NB_LINE:** the number of rows processed. This is an After variable and it returns an integer.  
A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  
To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.  
For further information about variables, see [Talend Studio User Guide](#). |

### Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as an input or as an intermediate component. It must be linked to an output component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitation</td>
<td>A JDK is required for this component to operate.</td>
</tr>
</tbody>
</table>
Scenario: Getting country names through a Web service

This scenario describes a three-component Job that uses a Web service to retrieve the country name corresponding to a given country code.

Linking components

Procedure

1. Drop the following components from the Palette onto the design workspace: **tFixedFlowInput**, **tWebService**, and **tLogRow**.

2. Connect the components together using Main Row links.

Configuring the input data

Procedure

1. Double-click **tFixedFlowInput** to open its Basic settings view and define the component properties.

2. Click the three-dot button next to the Edit schema field to open a dialog box where you can define the input schema.

3. In the open dialog box, click the plus button to add a column to the schema.

4. Click in the column and type in a name, CountryCode in this example.

5. Click OK to close the schema definition dialog box. The CountryCode column displays in the Values table in the component Basic settings view.

6. In the Values table, click in the Value column and enter the value of the CountryCode column, cn in this example. This country code will be passed to the tWebService component to retrieve the country name corresponding to it.
Configuring the Web service

Selecting the WSDL

Procedure

1. Double-click tWebService to open the component editor, or select the component in the design workspace and in the Basic settings view, click the three-dot button next to Service configuration.

2. In the WSDL field, enter the Web service address or browse to it, if the WSDL is locally stored, by clicking the [Browse...] button. In this example, the WSDL URL is http://www.webservicex.net/country.asmx?WSDL.

3. Click the refresh button next to the WSDL field to retrieve the WSDL description and display it in the fields that follow.

4. In the Port Name list, select the port you want to use, countrySoap in this example.

5. In the Operation list, select the service you want to use. In this example the selected service is GetCountryByCountryCode(parameters):string.

Configure the input mapping

Procedure

1. Click Next to open a new view in the editor.
2. In the panel to the right of the **Input mapping** view, select **[+] parameters** and then click the plus button on top to display the **[Parameter Tree]** dialog box where you can select any of the listed parameters.

The Web service in this example has only one input parameter, **CountryCode**.

**Note:**

If available, use the **Auto map!** button, located at the bottom left of the interface, to carry out the mapping operation automatically.

You now need to create a connection between the input schema and the input parameter of the defined Web service.

3. In the **Column** list, drag the column in the input schema you want to link to the input parameter of the Web service to the corresponding parameter in the panel to the right.

![Configuring the output mapping](image)

**Configuring the output mapping**

**Procedure**

1. Click **Next** to open a new view in the editor.
In the **Element** list to the left of the view, the output parameter of the web service displays automatically. However, you can add other parameters if you select [+] **parameters** and then click the plus button on top to display the [Parameter Tree] dialog box where you can select any of the parameters listed.

The Web service in this example has only one output parameter: `GetCountryByCountryCodeResult`.

2. In the panel to the right of the view, click the three-dot button next to **Edit Schema** to open a dialog box in which you can define the output schema.

3. In the **Output** list to the right of the dialog box, click the plus button to add a column to the output schema.

4. Click in the column and type in a name, **Result** in this example. This will retrieve the names of the country.

5. Click **OK** to validate your changes and to close the schema definition dialog box.

You now need to create a connection between the output parameter of the defined Web service and the schema of the output component.
6. In the **Element** list to the right of the editor, drag `parameters.GetCountryByCountryCodeResult` to the field that corresponds to the `Result` column to the right of the editor.

![Image of Talend Integration Suite - tWebService_1](image)

![Image of tWebService component](image)

**Note:**

If available, use the **Auto map** button, located at the bottom left of the interface, to carry out the mapping operation automatically.

7. Click **OK** to validate your changes and to close the editor.

8. Click the [...] button to open the [Schema] dialog box, and add a column to the output schema, `Result` in this example. Then click **OK** to validate the schema setting and close the dialog box.

9. Double-click the **tLogRow** component to open its **Basic settings** view, and click **Sync columns** to retrieve the schema from the proceeding component.

### Executing the Job

**Procedure**

Save your Job and press **F6** to execute it.
Results

The country name corresponding to the specified country code is returned from the Web service.
tWebServiceInput

Invokes a Method through a Web service.

This component calls the defined method from the invoked Web service, and returns the class as defined, based on the given parameters.

**tWebServiceInput Standard properties**

These properties are used to configure tWebServiceInput running in the Standard Job framework.

The Standard tWebServiceInput component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property type</th>
<th>Either <strong>Built-in</strong> or <strong>Repository</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in</strong>:</td>
<td>No property data stored centrally.</td>
</tr>
<tr>
<td><strong>Repository</strong>:</td>
<td>Select the Repository file where the properties are stored. The fields that follow are completed automatically using the data retrieved.</td>
</tr>
<tr>
<td>Click this icon...</td>
<td>Click this icon to open a WSDL schema wizard and store your WSDL connection in the <strong>Repository</strong> tree view. For more information about setting up and storing database connection parameters, see <em>Talend Studio User Guide</em>.</td>
</tr>
</tbody>
</table>

**Schema and Edit Schema**

A schema is a row description, it defines the number of fields to be processed and passed on to the next component. The schema is either **Built-in** or stored remotely in the **Repository**.

Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:

- **View schema**: choose this option to view the schema only.
- **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.
- **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the [Repository Content] window.

**Built-in**: You create the schema and store it locally for the relevant component. Related topic: see *Talend Studio User Guide*. |
<table>
<thead>
<tr>
<th><strong>WSDL</strong></th>
<th>Description of Web service bindings and configuration.</th>
</tr>
</thead>
</table>
| **Need authentication / Username and Password** | Select this check box and:  
- enter a username and a password in the corresponding fields if this is necessary to access the service. Or,  
- select the Windows authentication check box and enter the windows domain in the corresponding field if this is necessary to access the service.  
To enter the password, click the [...] button next to the password field, and then in the pop-up dialog box enter the password between double quotes and click OK to save the settings. |
| **Use http proxy** | Select this check box if you are using a proxy server and fill in the necessary information. |
| **Trust server with SSL** | Select this check box to validate the server certificate to the client via an SSL protocol and fill in the corresponding fields:  
**TrustStore file**: enter the path (including filename) to the certificate TrustStore file that contains the list of certificates that the client trusts.  
**TrustStore password**: enter the password used to check the integrity of the TrustStore data. |
| **Time out (second)** | Set a value in seconds for Web service connection time out. |
| **Method Name** | Enter the exact name of the Method to be invoked.  
The Method name MUST match the corresponding method described in the Web Service. The Method name is also case-sensitive. |
| **Parameters** | Enter the parameters expected and the sought values to be returned. Make sure that the parameters entered fully match the names and the case of the parameters described in the method. |
| **Advanced Use** | Select this check box to display the fields dedicated for the advanced use of tWebServiceInput:  
**WSDL2java**: click the three-dot button to generate Talend routines that hold the Java code necessary to connect and query the Web service.  
**Code**: replace the generated model Java code with the code necessary to connect and query the specified Web service using the code in the generated Talend routines. |
**Match Brackets**: select the number of brackets to be used to close the for loop based on the number of open brackets.

| tStatCatcher Statistics | Select this check box to gather the Job processing metadata at a Job level as well as at each component level. |

**Global Variables**

| Global Variables | **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box. |
| | **NB_LINE**: the number of rows processed. This is an After variable and it returns an integer. |
| | A Flow variable functions during the execution of a component while an After variable functions after the execution of the component. |
| | To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it. |
| | For further information about variables, see Talend Studio User Guide. |

**Usage**

| Usage rule | This component is generally used as a Start component. It must be linked to an output component. |

**Scenario: Getting country names through a Web service**

This scenario describes a two-component Job which uses a Web service method to obtain the country name corresponding to a given country code and displays the output on the Run console view.

**Procedure**

1. Drop a **tWebServiceInput** component and a **tLogRow** component from the Palette onto the design workspace.
2. Link the two components using a Row > Main connection.
3. Double-click the **tWebServiceInput** component to open its Basic settings view.
4. Click the [...] button next to Edit schema to open the [Schema] dialog box and define the schema for the response from the Web service.

   In this example, the schema is composed of only one column: Result.

5. In the WSDL field, specify the WSDL URL of the Web service to be used, http://www.webservicex.net/country.asmx?WSDL in this example.

6. If you are using a proxy server, select the Use http proxy check box and enter the necessary connection information.

7. In the Method name field, enter the method to be used. Note that the method name is case sensitive.

6. In this example, we use the method GetCountryByCountryCode.

8. In the Parameters area, click the plus [+ ] button to add a row to the table, then enter a country code, fr in this example.

9. Then save your Job and press F6 to execute it.
The country information is returned from the Web service and displayed on the Run view.
tWriteJSONField

Transforms the incoming data into JSON fields and transfers them to a file, a database table, etc.

Configuring a JSON Tree

When configuring a JSON tree, the default type of an element is string. If an element is not of type \textit{string}, you need to add an attribute for the element to set its type.

- For an \textit{integer}, \textit{double}, \textit{float}, or \textit{boolean} element, you need to add an attribute named \texttt{type} and set its static value to \texttt{integer}, \texttt{number}, \texttt{float}, or \texttt{boolean} respectively.

- For an \textit{array} element, you need to add an attribute named \texttt{class} and set its static value to \texttt{array}, and then add a sub-element named \texttt{element} set as a loop.

- For an \textit{object} element, you need to add an attribute named \texttt{class} and set its static value to \texttt{object}.

The following figure shows an example of JSON tree configuration:

![JSON Tree Configuration Example]

\textbf{tWriteJSONField Standard properties}

These properties are used to configure tWriteJSONField running in the Standard Job framework. The Standard tWriteJSONField component belongs to the Processing family. The component in this framework is generally available.
## Basic settings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output Column</strong></td>
<td>List of the columns defined in the output schema to hold the JSON field generated.</td>
</tr>
<tr>
<td><strong>Configure JSON Tree</strong></td>
<td>Opens the interface to create the JSON data structure. For more information, see <em>Configuring a JSON Tree</em> on page 4040.</td>
</tr>
</tbody>
</table>
| **Schema and Edit Schema**      | A schema is a row description. It defines the number of fields (columns) to be processed and passed on to the next component. When you create a Spark Job, avoid the reserved word `line` when naming the fields.  
 Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  - **View schema**: choose this option to view the schema only.  
  - **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  - **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window. |
| **Sync columns**                | Click to synchronize the output file schema with the input file schema. The Sync function only displays once the Row connection is linked with the output component. |
| **Group by**                    | Define the aggregation set, the columns you want to use to regroup the data. **Warning**: Make sure that the data to be grouped is in sequential order. |
| **Remove root node**            | Select this check box to remove the root node from the JSON field generated.  |

**Built-In**: You create and store the schema locally for this component only. Related topic: see *Talend Studio User Guide*.

**Repository**: You have already created the schema and stored it in the Repository. You can reuse it in various projects and Job designs. Related topic: see *Talend Studio User Guide*.
Advanced settings

**Quote all values**
- Select this check box to surround all values including number and boolean values in the generated JSON field with double quotation marks.
- Clear this check box to surround only string values in the generated JSON field with double quotation marks.

**tStatCatcher Statistics**
- Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

Global Variables

**Global Variables**

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.
- **NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.
- A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.
- To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
- For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**
- Preceded by an input component, this component wraps the incoming data into a JSON field.

Scenario: Writing flat data into JSON fields

In this scenario, flat data is wrapped into JSON fields via **tWriteJSONField**.

Linking the components

**Procedure**

1. Drop the following components from the **Palette** onto the design workspace: **tFixedFlowInput**, **tWriteJSONField** and **tLogRow**.
2. Link **tFixedFlowInput** and **tWriteJSONField** using a **Row > Main** connection.
3. Link **tWriteJSONField** and **tLogRow** using a **Row > Main** connection.
Configuring the components

Procedure

1. Double-click tFixedFlowInput to display its Basic settings view.

2. Click Edit schema to open the schema editor.

   Click the [+] button to add three columns, namely firstname, lastname and dept, with the type of string.
   Click OK to close the editor.

3. Select the Use Inline Content option and enter the data below in the Content box:

   Andrew;Wallace;Doc
   John;Smith;R&D
   Christian;Dior;Sales

4. Click tWriteJSONField to display its Basic settings view.
Select the **Remove root node** option to remove the root node setting from the JSON fields generated.

5. Click **Configure JSON Tree** to open the XML tree editor.

The schema of `tFixedFlowInput` appears in the **Linker source** panel.

6. In the **Linker target** panel, click the default **rootTag** and type in **staff**, which is the root node of the JSON field to be generated.

7. Right-click **staff** and select **Add Sub-element** from the context menu.

8. In the pop-up box, enter the sub-node name, namely **firstname**.
Repeat the steps to add two more sub-nodes, namely `lastname` and `dept`.

9. Right-click `firstname` and select **Set As Loop Element** from the context menu.

10. Drop `firstname` from the **Linker source** panel to its counterpart in the **Linker target** panel.
    In the pop-up dialog box, select **Add linker to target node**.

Click **OK** to close the dialog box.
Repeat the steps to link the two other items.
Click **OK** to close the XML tree editor.

11. Click **Edit schema** to open the schema editor.

12. Click the [+] button in the right panel to add one column, namely `staff`, which will hold the JSON data generated.
    Click **OK** to close the editor.

13. Double-click **tLogRow** to display its **Basic settings** view.
Executing the Job

Procedure

1. Press Ctrl + S to save the Job.
2. Click F6 to execute the Job.

As shown above, the JSON fields have been generated correctly, with the root node settings removed.

Related Scenarios

For related scenarios, see:

- Scenario 1: Retrieving error messages while extracting data from JSON fields on page 949.
- Scenario: Extracting the structure of an XML file and inserting it into the fields of a database table on page 4049.
- Mapping XML data on page 113.
tWriteXMLField

Reads an input XML file and extracts the structure to insert it in defined fields of the output XML file.

**tWriteXMLField Standard properties**

These properties are used to configure tWriteXMLField running in the Standard Job framework.

The Standard tWriteXMLField component belongs to the XML family.

The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th><strong>Output Column</strong></th>
<th>Select the destination field in the output component where you want to write the XML structure.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configure XML Tree</strong></td>
<td>Opens the interface that supports the creation of the XML structure you want to write in a field. For more information about the interface, see Defining the XML tree on page 111.</td>
</tr>
</tbody>
</table>
| **Schema and Edit Schema** | A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either built-in or remote in the Repository. Click **Edit schema** to make changes to the schema. If the current schema is of the **Repository** type, three options are available:  
  • **View schema**: choose this option to view the schema only.  
  • **Change to built-in property**: choose this option to change the schema to **Built-in** for local changes.  
  • **Update repository connection**: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select **No** upon completion and choose this schema metadata again in the **Repository Content** window.  

| **Sync columns** | Click to synchronize the output file schema with the input file schema. The Sync function only displays once the Row connection is linked with the input component. |

**Built-in**: You create the schema and store it locally for this component only. Related topic: see Talend Studio User Guide.

**Repository**: You already created the schema and stored it in the Repository, hence can be reused in various projects and job flowcharts. Related topic: see Talend Studio User Guide.
<table>
<thead>
<tr>
<th><strong>Group by</strong></th>
<th>Define the aggregation set, the columns you want to use to regroup the data.</th>
</tr>
</thead>
</table>

### Advanced settings

<table>
<thead>
<tr>
<th><strong>Remove the XML declaration</strong></th>
<th>Select this check box if you do not want to include the XML header.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create empty element if needed</strong></td>
<td>This check box is selected by default. If the Related Column in the XML tree editor has null values, or if no column is associated with the XML node, this option creates an open/close tag in the expected place.</td>
</tr>
<tr>
<td><strong>Expand Empty Element if needed (for dom4j)</strong></td>
<td>Select this option to allow a null element to appear in the form of tag pair, e.g. <code>&lt;element&gt;&lt;/element&gt;</code>. Otherwise, such an element appears as a solo tag, e.g. <code>&lt;element/&gt;</code>. For more information about XML tags, see <a href="http://www.tizag.com/xmlTutorial/xmltag.php">http://www.tizag.com/xmlTutorial/xmltag.php</a>.</td>
</tr>
<tr>
<td><strong>Create associated XSD file</strong></td>
<td>If one of the XML elements is defined as a Namespace element, this option will create the corresponding XSD file.</td>
</tr>
<tr>
<td><strong>Advanced separator (for number)</strong></td>
<td>Select this check box if you want to modify the separators used by default for numbers. <strong>Thousands separator</strong>: enter between brackets the separators to use for thousands. <strong>Decimal separator</strong>: enter between brackets the separators to use for decimals.</td>
</tr>
<tr>
<td><strong>Generation mode</strong></td>
<td>Select the appropriate generation mode according to your memory availability. The available modes are:</td>
</tr>
<tr>
<td></td>
<td>• Slow and memory-consuming (Dom4j)</td>
</tr>
<tr>
<td></td>
<td>• Fast with low memory consumption</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>Select the encoding type in the list or select Custom and define it manually. This field is compulsory when working with databases.</td>
</tr>
</tbody>
</table>
tStatCatcher Statistics

Select this check box to gather the Job processing metadata at a Job level as well as at each component level.

Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

**NB_LINE**: the number of rows read by an input component or transferred to an output component. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.

Usage

**Usage rule**

This component can be used as intermediate step in a data flow.

Scenario: Extracting the structure of an XML file and inserting it into the fields of a database table

This three-component scenario allows to read an XML file, extract the XML structure, and finally outputs the structure to the fields of a database table.

Procedure

**Procedure**

1. Drop the following components from the Palette onto the design workspace: **tFileInputXml**, **tWriteXMLField**, and **tMysqlOutput**.

   Connect the three components using **Main links**.

2. Double-click **tFileInputXml** to open its **Basic settings** view and define its properties.
3. If you have already stored the input schema in the Repository tree view, select Repository first from the Property Type list and then from the Schema list to display the [Repository Content] dialog box where you can select the relevant metadata. For more information about storing schema metadata in the Repository tree view, see Talend Studio User Guide.

4. If you have not stored the input schema locally, select Built-in in the Property Type and Schema fields and fill in the fields that follow manually. For more information about tFileInputXML properties, see tFileInputXML on page 1089. If you have selected Built-in, click the three-dot button next to the Edit schema field to open a dialog box where you can manually define the structure of your file.

5. In the Look Xpath query field, enter the node of the structure where the loop is based. In this example, the loop is based on the customer node. Column in the Mapping table will be automatically populated with the defined file content.

In the Xpath query column, enter between inverted commas the node of the XML file that holds the data corresponding to each of the Column fields.

6. In the design workspace, click tWriteXMLField and then in the Component view, click Basic settings to open the relevant view where you can define the component properties.

7. Click the three-dot button next to the Edit schema field to open a dialog box where you can add a line by clicking the plus button.
8. Click in the line and enter the name of the output column where you want to write the XML content, *CustomerDetails* in this example.
Define the type and length in the corresponding fields, *String* and 255 in this example.
Click *Ok* to validate your output schema and close the dialog box.
In the **Basic settings** view and from the **Output Column** list, select the column you already defined where you want to write the XML content.

9. Click the three-dot button next to **Configure Xml Tree** to open the interface that helps to create the XML structure.

10. In the **Link Target** area, click *rootTag* and rename it as *CustomerDetails*.
In the **Linker source** area, drop *CustomerName* and *CustomerAddress* to *CustomerDetails*. A dialog box displays asking what type of operation you want to do.
Select *Create as sub-element of target node* to create a sub-element of the *CustomerDetails node*.
Right-click *CustomerName* and select from the contextual menu **Set As Loop Element**.
Click *OK* to validate the XML structure you defined.

11. Double-click **tMysqlOutput** to open its **Basic settings** view and define its properties.
If you have already stored the schema in the DB Connection node in the Repository tree view, select Repository from the Schema list to display the [Repository Content] dialog box where you can select the relevant metadata.

For more information about storing schema metadata in the Repository tree view, see Talend Studio User Guide.

If you have not stored the schema locally, select Built-in in the Property Type and Schema fields and enter the database connection and data structure information manually. For more information about tMysqlOutput properties, see tMysqlOutput on page 2515.

In the Table field, enter the name of the database table to be created, where you want to write the extracted XML data.

From the Action on table list, select Create table to create the defined table.

From the Action on data list, select Insert to write the data.

Click Sync columns to retrieve the schema from the preceding component. You can click the three-dot button next to Edit schema to view the schema.

Save your Job and click F6 to execute it.

Results

<table>
<thead>
<tr>
<th>CustomerDetails</th>
</tr>
</thead>
</table>

The tWriteXMLField fills every field of the CustomerDetails column with the XML structure of the input file: the XML processing instruction <!--xml version="1.0" encoding="ISO-8859-15"--> , the first node that separates each client <CustomerDetails> and finally customer information <CustomerAddress> and <CustomerName>. 

4052
tXMLMap

Transforms and routes data from single or multiple sources to single or multiple destinations.

� tXMLMap is an advanced component fine-tuned for transforming and routing XML data flow (data of the Document type), especially when processing numerous XML data sources, with or without flat data to be joined.

tXMLMap Standard properties

These properties are used to configure tXMLMap running in the Standard Job framework.

The Standard tXMLMap component belongs to the Processing and the XML families.

The component in this framework is generally available.

Basic settings

<table>
<thead>
<tr>
<th>Map Editor</th>
<th>It allows you to define the tXMLMap routing and transformation properties.</th>
</tr>
</thead>
</table>

Advanced settings

<table>
<thead>
<tr>
<th>tStatCatcher Statistics</th>
<th>Select this check box to gather the Job processing metadata at the Job level as well as at each component level.</th>
</tr>
</thead>
</table>

Global Variables

| Global Variables | ERROR_MESSAGE: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.  

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.  

To fill up a field or expression with a variable, press Ctrl + Space to access the variable list and choose the variable to use from it.  

For further information about variables, see Talend Studio User Guide. |

Usage

| Usage rule | Possible uses are from a simple reorganization of fields to the most complex jobs of data multiplexing or demultiplexing transformation, concatenation, inversion, filtering and so on.  

When needs be, you can define sophisticated outputting strategy for the output XML flows using group element. |
|------------|-------------------------------------------------------------------------------------------------|
aggregate element, empty element and many other features such as All in one. For further information about these features, see Talend Studio User Guide.

It is used as an intermediate component and fits perfectly the process requiring many XML data sources, such as, the ESB request-response processes.

| Limitation          | The limitations to be kept in mind are:
|---------------------|--------------------------------------------------
|                     | - The use of this component supposes minimum Java and XML knowledge in order to fully exploit its functionalities. |
|                     | - This component is a junction step, and for this reason it cannot be a start nor an end component in the Job. |
|                     | - At least one loop element is required for each XML data flow involved. |

The following sections present several generic use cases about how to use the tXMLMap component, while if you need some specific examples using this component along with the ESB components to build data services, see the scenarios for the ESB components:

- Scenario 2: Using tESBConsumer with custom SOAP Headers on page 833
- Scenario: Requesting airport names based on country codes on page 845
- Scenario: Service sending a message without expecting a response on page 860
- Scenario: Returning Hello world response on page 871

**Mapping and transforming XML data**

The following scenario creates a three-component Job that maps and transforms data from an XML source file Customer.xml, and generates an XML output flow which could be reused for various purposes, such as for an ESB request, in the future based on the XML tree structure of the file Customer_State.xml.

These three components are:

- **tFileInputXML**: provides the input data to tXMLMap.
- **tXMLMap**: maps and transforms the received XML data flows into one single XML data flow.
- **tLogRow**: displays the output data.

The content of the XML file Customer.xml is as follows:

```xml
<?xml version="1.0" encoding="ISO-8859-15"?>
<Customers>
    <Customer RegisterTime="2001-01-17 06:26:40.000">
        <Name />
        <id>1</id>
    </Customer>
</Customers>
```
<CustomerName>Griffith Paving and Sealcoatin</CustomerName>
</Name>
<Address>
<br_customer_address>talend@apres91</br_customer_address>
</Address>
<Revenue>
<br_sum>67852</br_sum>
</Revenue>
</Customer>
<Customer RegisterTime="2002-06-07 09:40:00.000">
<Name>
<br_id>2</br_id>
</Name>
<Address>
<br_customer_address>511 Maple Ave. Apt. 1B</br_customer_address>
</Address>
<Revenue>
<br_sum>88792</br_sum>
</Revenue>
</Customer>
<Customer RegisterTime="1987-02-23 17:33:20.000">
<Name>
<br_id>3</br_id>
</Name>
<Address>
<br_customer_address>1859 Green Bay Rd.</br_customer_address>
</Address>
<Revenue>
<br_sum>1225</br_sum>
</Revenue>
</Customer>
<Customer RegisterTime="1992-04-28 23:26:40.000">
<Name>
<br_id>4</br_id>
</Name>
<Address>
<br_customer_address>456 Grossman Ln.</br_customer_address>
</Address>
<Revenue>
<br_sum>64493</br_sum>
</Revenue>
</Customer>
</Customers>

The content of the XML file Customer_State.xml is as follows:

```xml
<?xml version="1.0" encoding="ISO-8859-15"?>
<customers>
  <customer id="1">
    <CustomerName>Griffith Paving and Sealcoatin</CustomerName>
    <CustomerAddress>talend@apres91</CustomerAddress>
    <idState>2</idState>
  </customer>
  <customer id="2">
    <CustomerName>Bill's Dive Shop</CustomerName>
    <CustomerAddress>511 Maple Ave. Apt. 1B</CustomerAddress>
    <idState>3</idState>
  </customer>
  <customer id="3">
    <CustomerName>Glenn Oaks Office Supplies</CustomerName>
    <CustomerAddress>1859 Green Bay Rd.</CustomerAddress>
    <idState>2</idState>
  </customer>
  <customer id="4">
    <CustomerName>DBN Bank</CustomerName>
    <CustomerAddress>456 Grossman Ln.</CustomerAddress>
    <idState>3</idState>
  </customer>
</customers>
```
**Adding and linking the components**

**Procedure**

1. Create a new Job and add a `tFileInputXML` component, a `tXMLMap` component, a `tLogRow` component by typing their names in the design workspace or dropping them from the Palette.

2. Label the `tFileInputXML` component *Customers* to better identify its function.

   **Note:**
   
   A component used in the workspace can be labelled the way you need. For further information about how to label a component, see *Talend Studio User Guide*.

3. Link the `tFileInputXML` component labelled *Customers* to the `tXMLMap` component using a *Row > Main* connection.

4. Link the `tXMLMap` component to the `tLogRow` component using a *Row > *New Output* (Main)* connection. In the pop-up dialog box, enter the name of the output connection, *Customer* in this scenario.

   ![tXMLMap_1Output](image)

**Configuring the input flow**

**Procedure**

1. Double-click the `tFileInputXML` component labelled *Customers* to open its *Basic settings* view.

   ![Customers(tFileInputXML_1)](image)

2. Click the [...] button next to *Edit schema* and in the *Schema* dialog box define the schema by adding one column *Customer* of *Document* type.
Note that the Document data type is essential for making full use of tXMLMap. For further information about this data type, see Talend Studio User Guide.

3. Click OK to validate the changes and close the dialog box. One row is added automatically to the Mapping table.

4. In the File name/Stream field, browse to or type in between double quotation marks the path to the XML source file that provides the customer data. In this scenario, it is E:/Customer.xml.

5. In the Loop XPath query field, type in an XPath expression between double quotation marks to specify the node on which the loop is based. In this scenario, it is /, which means to perform look query from the root.

6. In the XPath query column of the Mapping table, type in the fields to be queried between double quotation marks. In this scenario, it is ., which means all fields under the current node (root) will be extracted.

7. In the Get Nodes column of the Mapping table, select the check box. In order to build the Document type data flow, it is necessary to get the nodes from this component.

**Configuring tXMLMap for transformation**

**Procedure**

1. Double-click the tXMLMap component to open its Map Editor.
Note that the input area is already filled with the default basic XML structure and the top table is the main input table.

2. In the \textit{row1} input table, right-click the \textit{Customer} node and from the contextual menu select \textbf{Import From File}. In the pop-up dialog box, browse to the XML source file to import therefrom the XML tree structure used by the data to be received by \texttt{tXMLMap}. In this scenario, the XML source file is \textit{Customer.xml}, which is the input data to the \texttt{tFileInputXML} component labelled \textit{Customers}.

\begin{itemize}
\item \textbf{Note:}
\begin{itemize}
\item You can also import an XML tree from an XSD file. When importing either an input or an output XML tree structure from an XSD file, you can choose an element as the root of your XML tree. For more information on importing an XML tree structure from an XSD file, see \textit{Talend Studio User Guide}.
\end{itemize}
\end{itemize}

3. In the imported XML tree, right-click the \textit{Customer} node and from the contextual menu select \textbf{As loop element} to set it as the loop element.

4. On the lower part of this map editor, click the \textbf{Schema editor} tab to display the corresponding view. Then on the right side of this view, add one column \textit{Customer_States of Document} type to the \textit{Customer} schema table. The corresponding XML root is added automatically to the \textit{Customer} output table on the top right side which represents the output flow.

5. In the \textit{Customer} output table, right-click the \textit{Customer_States} node and from the contextual menu select \textbf{Import From File}. In the pop-up dialog box, browse to the XML file from which the XML tree structure is imported. In this scenario, it is \textit{Customer_State.xml}. 
6. Right-click the `customer` node and from the contextual menu select **As loop element** to set it as the loop element.

7. In the `row1` input table, click the `id` node and drop it to the **Expression** column in the row of the `@id` node in the `Customer` output table.

Do the same to map `CustomerName` to `CustomerName`, `CustomerAddress` to `CustomerAddress`, and `idState` to `idState` from the input table to the output table.

**Note:**

In some circumstances, you may have to keep empty elements in your output XML tree. If so, you can use **tXMLMap** to manage them. For further information about how to manage empty elements using **tXMLMap**, see *Talend Studio User Guide*.

8. On the top of the `Customer` output table, click the wrench icon and set the value of the **All in one** property to **true** to generate a single XML flow. For further information about the **All in one** feature, see *Talend Studio User Guide*. 
9. Click OK to validate the changes and close the Map Editor.

   Note:
   If you close the Map Editor without having set the required loop elements as described earlier in this scenario, the root element will be automatically set as the loop element.

**Configuring tLogRow to display the customer information**

**Procedure**

1. Double-click the tLogRow component to open its Basic settings view.
2. Click the Sync columns button to retrieve the schema from its preceding component.

**Saving and executing the Job**

**Procedure**

1. Press Ctrl+S to save the Job.
2. Press F6 to execute the Job.

```xml
<customers>
  <customer id="1" CustomerName="Griffith Paving and Sealcoating" CustomerAddress="talex@apress91"/>
  <customer id="2" CustomerName="Bill's Dive Shop" CustomerAddress="511 Maple Ave. Apt 1B"/>
  <customer id="3" CustomerName="Glenn Oaks Office Supplies" CustomerAddress="1859 Green Bay Rd"/>
  <customer id="4" CustomerName="DBN Bank" CustomerAddress="456 Grossman Ln"/>
</customers>
```

As shown above, the transformed customer information is displayed on the console.

**Launching a lookup flow to join complementary data**

Based on the previous scenario, this scenario shows how to use a lookup flow to join data of interest in the XML file `USState.xml` to the main flow. Another tFileInputXML component is added to the Job to load data from the lookup file `USState.xml` to the processing component tXMLMap.
The content of the XML file USState.xml is as follows:

```xml
<?xml version="1.0" encoding="ISO-8859-15"?>
<USStates>
  <States>
    <idState>1</idState>
    <LabelState>Alabama</LabelState>
  </States>
  <States>
    <idState>2</idState>
    <LabelState>Connecticut</LabelState>
  </States>
  <States>
    <idState>3</idState>
    <LabelState>Ohio</LabelState>
  </States>
  <States>
    <idState>4</idState>
    <LabelState>Wyoming</LabelState>
  </States>
  <States>
    <idState>5</idState>
    <LabelState>Hawaii</LabelState>
  </States>
</USStates>
```

Adding and linking another input component

Procedure
1. In your Studio, open the Job used in the previous scenario to display it in the design workspace.
2. Add another tFileInputXML component to the Job by typing its name in the design workspace or dropping it from the Palette. Label the component USStates to better identify its function.
3. Link the tFileInputXML component labelled USStates to the tXMLMap component using a Row > Main connection, and the connection is automatically changed to a lookup flow.

Configuring the input flow for lookup

Procedure
1. Double-click the tFileInputXML component labelled USStates to open its Basic settings view.
2. Click the [...] button next to Edit schema and in the [Schema] dialog box define the schema by adding one column USState of Document type.

3. Click OK to validate the changes and close the dialog box. One row is added automatically to the Mapping table.

4. In the File name/Stream field, browse to or type in between double quotation marks the path to the XML source file that holds the complementary data. In this scenario, it is E:/USState.xml.

5. In the Loop XPath query field, type in an XPath expression between double quotation marks to specify the node on which the loop is based. In this scenario, it is /, which means to perform look query from the root.

6. In the XPath query column of the Mapping table, type in the fields to be queried between double quotation marks. In this scenario, it is ., which means all fields under the current node (root) will be extracted.

7. In the Get Nodes column of the Mapping table, select the check box. This retrieves the XML structure for the Document type data.

Configuring tXMLMap for transformation

Procedure

1. Double-click the tXMLMap component to open its Map Editor.
2. In the row2 input table, right-click the USState node and from the contextual menu select **Import From File**. In the pop-up dialog box, browse to the XML source file to import therefrom the XML tree structure used by the data to be received by **tXMLMap**. In this scenario, the XML source file is **USState.xml**, which is the input data to **tFileInputXML** labelled **USStates**.

3. In the imported XML tree, right-click the States node and from the contextual menu select **As loop element** to set it as the loop element.
4. In the row1 main input table, click the idState node and drop it to the Exp.key column in the row of the idState node in the row2 lookup input table. This creates a join between the two input tables on the idState data, among which the idState node from the main flow provides the lookup key.

5. In the row2 lookup input table, click the LabelState node and drop it on the customer node in the Customer output table. A dialog box pops up.

6. In the pop-up dialog box, select Create as sub-element of target node and click OK. A new LabelState sub-element is added to the output XML tree and mapped with the LabelState node in the lookup input table.
7. Click OK to validate the mappings and close the Map Editor.

Saving and executing the Job

Procedure

1. Press Ctrl+S to save the Job.
2. Press F6 to run the Job.

As shown above, the state names from the lookup file with the state IDs matching those in the main input file are added to the data flow and the combined information is displayed on the console.

Results

A step-by-step tutorial related to this Join topic is available on the Talend Technical Community Site. For further information, see http://talendforge.org/tutorials/tutorial.php?language=english&idTuto=101.
Mapping data using a filter

Based on Launching a lookup flow to join complementary data on page 4060, this scenario presents how to apply filter condition(s) to select the data of interest using tXMLMap.

**Mapping data using a filter**

**Procedure**

1. In your Studio, open the Job used in the previous scenario to display it in the design workspace.

2. Double-click the tXMLMap component to open its Map Editor.

3. On the top of the Customer output table, click the button to open the filter area.
4. Drop the `idState` node in the main input table to the filter area. The XPath `[@1.Customer:/Customers/Customer/Address/idState]` of the `idState` node is added automatically to this filter area. Enter `== 2` after the XPath of the `idState` node, and the complete filter condition becomes `[row1.Customer:/Customers/Customer/Address/idState] == 2`. This means only the customer data with the state id of 2 will be passed to the output flow.

5. Click OK to validate the changes and close the map editor.

6. Press Ctrl + S to save the Job and then F6 to run the Job.

Results

As shown above, the customers *Griffith Paving and Sealcoating* and *Glenn Oaks Office Supplies*, whose state id is 2 are displayed on the console.

Catching the data rejected by lookup and filter

The data rejected by the lookup and filter conditions set in tXMLMap can be caught and outputted by this component itself.
Based on Mapping data using a filter on page 4066, this scenario presents how to catch the data rejected by the lookup and the filter set up in the previous scenarios. Another `tLogRow` component is added to the Job used in the previous scenario to display the rejected data.

**Adding and linking another output component**

**Procedure**

1. In your Studio, open the Job used in the previous scenario to display it in the design workspace.
2. Add another `tLogRow` component to the Job by typing its name in the design workspace or dropping it from the Palette.
3. Link the `tXMLMap` component to the second `tLogRow` using a Row > "New Output" (Main) connection. In the pop-up dialog box, enter the name of the output connection, `Reject` in this example.
Configuring tXMLMap for transformation

Procedure

1. Double-click the tXMLMap component to open its Map Editor. An empty Reject output table that carries the rejected data has been added to the output side to represent the output data flow carrying the rejected data.

2. In the row1 main input table, click the id node and drop it on the Reject output table. A column id is added to the Reject schema table in the Schema editor on the lower part of the map editor.

3. Do the same to drop CustomerName, CustomerAddress, and idState in the row1 main input table and LabelState in the row2 lookup input table on the Reject output table. Another four columns CustomerName, CustomerAddress, idState, and LabelState are added to the Reject schema table in the Schema editor.

   Note:
   In this scenario, the Reject output flow uses the flat data type. However, you can create an XML tree view for this flow similar to the Customer output flow using the Document data type. For further information about how to use the Document type, see Mapping and transforming XML data on page 4054.

4. On the top of the Reject output table, click the button to open the property setting area.
5. Set the value of the **Catch Output Reject** property to **true** to catch the data rejected by the filter set up in the previous scenario for the **Customer** output flow.

6. Set the value of the **Catch Lookup Inner Join Reject** property to **true** to catch the data rejected by the inner join operation.

7. Click **OK** to validate the changes and close the map editor.

### Configuring the output flow

**Procedure**

1. Double-click the second **tLogRow** component to open its **Basic settings** view.

2. Click the **Sync columns** button to retrieve the schema from its preceding component.

3. In the **Mode** area, select **Table (print values in cells of a table)** for better readability of the result.

### Saving and executing the Job

**Procedure**

1. Press **Ctrl+S** to save the Job.

2. Press **F6** to run the Job.

The captured data rejected by the filter and the lookup reads as follows in the **Run** view:

```xml
[statistics] connecting to socket on port 4058
[statistics] connected
<customer><customer id="1"><CustomerName>Griffith Paving and Seals</CustomerName><CustomerAddress>telend@apres91</CustomerAddress><idState>2</idState><LabelState>Connecticut</LabelState><customer><customer id="3"><CustomerName>Glenn Oaks Office Supplies</CustomerName><CustomerAddress>1859 Green Bay Rd</CustomerAddress><idState>2</idState><LabelState>Connecticut</LabelState><customer></customer></customers>
```

<table>
<thead>
<tr>
<th>tLogRow_2</th>
<th>id</th>
<th>CustomerName</th>
<th>CustomerAddress</th>
<th>idState</th>
<th>LabelState</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>[Bill’s Dive Shop]</td>
<td>511 Maple Ave. Apt. 1E</td>
<td>1</td>
<td>Ohio</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>[WBB Bank]</td>
<td>456 Greenman Ln.</td>
<td>3</td>
<td>Ohio</td>
</tr>
</tbody>
</table>

[statistics] disconnected
As shown above, the data whose idState value is 2 is selected by the filter set up in the previous scenario and displayed in the upper part, and the data whose idState value is not 2 is rejected and displayed in the lower part.

**Mapping data using a group element**

Based on Launching a lookup flow to join complementary data on page 4060, this scenario presents how to set up an element as group element in the Map Editor of tXMLMap to group the output data. For more information about how to group the output data using tXMLMap, see Talend Studio User Guide.

The objective of this scenario is to group the customer id and the customer name information according to the states the customers come from. You need to reconstruct the XML tree view of the Customer output table by considering the following factors:

- The elements tagging the customer id and the customer name information should be located under the loop element. Thus they are the sub-elements of the loop element.
- The loop element and its sub-elements should be dependent directly on the group element.
- The element tagging the state information used as the grouping condition should be dependent directly on the group element.
- The group element cannot be the root element.

Based on this analysis, the XML structure of the output data should read as follows. The customers node is the root element, the customer node is set as the group element and the output data is grouped according to the LabelState element.
To put a group element into effect, the XML data to be processed should have been sorted, for example via your XML tools, around the element that will be used as the grouping condition. In this example, the customers possessing the same state id should be put together. The input data in the XML file *Customer.xml* should read as follows:

```xml
<?xml version="1.0" encoding="ISO-8859-15"?>
<Customers>
  <Customer RegisterTime="2001-01-17 06:26:40.000">
    <Name>
      <id>1</id>
      <CustomerName>Griffith Paving and Sealcoatin</CustomerName>
    </Name>
    <Address>
      <CustomerAddress>talend@apres91</CustomerAddress>
      <idState>2</idState>
    </Address>
    <Revenue>
      <Sum1>67852</Sum1>
    </Revenue>
  </Customer>
  <Customer RegisterTime="1987-02-23 17:33:20.000">
    <Name>
      <id>3</id>
      <CustomerName>Glenn Oaks Office Supplies</CustomerName>
    </Name>
    <Address>
      <CustomerAddress>1859 Green Bay Rd.</CustomerAddress>
      <idState>2</idState>
    </Address>
    <Revenue>
      <Sum1>1225</Sum1>
    </Revenue>
  </Customer>
  <Customer RegisterTime="2002-06-07 09:40:00.000">
    <Name>
      <id>2</id>
      <CustomerName>Bill's Dive Shop</CustomerName>
    </Name>
    <Address>
      <CustomerAddress>511 Maple Ave. Apt. 1B</CustomerAddress>
      <idState>3</idState>
    </Address>
    <Revenue>
      <Sum1>88792</Sum1>
    </Revenue>
  </Customer>
  <Customer RegisterTime="1992-04-28 23:26:40.000">
    <Name>
      <id>4</id>
      <CustomerName>DBN Bank</CustomerName>
    </Name>
    <Address>
      <CustomerAddress>456 Grossman Ln.</CustomerAddress>
      <idState>3</idState>
    </Address>
    <Revenue>
      <Sum1>64493</Sum1>
    </Revenue>
  </Customer>
</Customers>
```
**Mapping data using a group element**

**Procedure**

1. In your Studio, open the Job used in Launching a lookup flow to join complementary data on page 4060 to display it in the design workspace, and double-click the tXMLMap component to open its Map Editor.

2. In the XML tree view of the Customer output table, right-click the customer (loop) node and select **Delete** from the contextual menu. Thus all of the elements under the customers root node are removed, then you can reconstruct the XML tree view that can be used to group the output data of interest.

3. Right-click the customers root node and select **Create Sub-Element** from the contextual menu. In the pop-up dialog box, enter the name of the new sub-element. In this example, it is customer.

4. Click **OK** to validate the changes and close the dialog box. A customer node is added under the customers root node in the output table.

5. In the row2 lookup input table, select the LabelState node and drop it onto the customer node in the output table. In the pop-up dialog box, select **Create as sub-element of target node** and click **OK** to close the dialog box. A LabelState node is added under the customer node in the output table.

6. Right-click the customer node in the output table and select **Create Sub-Element** from the contextual menu. In the pop-up dialog box, enter the name of the new sub-element. In this example, it is Name.

7. Click **OK** to validate the changes and close the dialog box. A Name node is added under the customer node in the output table.

8. In the row1 main input table, select the id and CustomerName nodes and drop them onto the Name node in the output table. In the pop-up dialog box, select **Create as sub-element of target node** and click **OK** to close the dialog box. A id node and a CustomerName node are added under the Name node in the output table.

9. In the output table, right-click the Name node and from the contextual menu select **As loop element** to set it as the loop element, then right-click the customer node and from the contextual menu select **As group element** to group the output data according to the LabelState element.
10. Click OK to validate the changes and close the map editor.
11. Press Ctrl+S to save the Job and then F6 to run the Job.

Results
As shown above, the id element and the CustomerName element contained in the loop are grouped according to the LabelState element. The group element customer tags the start and the end of each group.

Classifying the output data with aggregate element

Based on the previous scenarios, this scenario presents how to set up an element as aggregate element in the Map Editor of tXMLMap to classify the output data into separate XML flows. For more information about how to aggregate the output data using tXMLMap, see Talend Studio User Guide.

The objective of this scenario is to classify the customer id and the customer name information using an aggregate element in accordance with the states they come from and then to send these classes separately in different XML flows to the next component.

To put an aggregate element into effect, the XML data to be processed should have been sorted, for example via your XML tools, around the element that will be used as the aggregating condition. In
this example, the customers possessing the same state id should be put together. The input data in the XML file Customer.xml should be same as the input data in.

Classifying the output data with aggregate element

Procedure

1. In your Studio, open the Job used in, and double-click the tXMLMap component to open its Map Editor.
2. Right-click the customer element in the output table and select Remove group element from the contextual menu.
3. On the top of the output table, click the wrench icon and set the value of the All in one property to false.
4. Right-click the LabelState element and select As aggregate element from the contextual menu. The LabelState element tags the state information of each customer and the customer information will be classified according to the state information.

To make the aggregate element available, ensure that the value of the All in one property is set to false. For further information about the All in one feature, see Talend Studio User Guide.

5. Click OK to validate the changes and close the Map Editor.
6. Press Ctrl+S to save the Job and then F6 to run the Job.
Results

As shown above, tXMLMap outputs two separate XML flows, each of which carries the information of one state and the customers from that state.

Restructuring products data using multiple loop elements

The following scenario creates a four-component Job that restructures the products data from an XML source file ProductsIn.xml using multiple loop elements.

These four components are:

- **tFileInputXML**: reads the source products data and passes it to the tXMLMap component.
- **tXMLMap**: transforms the input flow to the expected structure streamlined.
- **tLogRow**: presents the execution result on the console.
- **tFileOutputXML**: writes the output flow into an XML file.

The content of the source XML file ProductsIn.xml is as follows:

```xml
<?xml version="1.0" encoding="ISO-8859-15"?>
<products category="1" name="laptop">
  <!-- Summary -->
  <summary>
    <company>DELL, HP</company>
    <sales unit="Dollars">12345678910.12345</sales>
    <model>business</model>
  </summary>
  <!-- Loop1 manufacture -->
  <manufacture id="manu_1" date="2012-10-30">
    <name>DELL</name>
  </manufacture>
  <manufacture id="manu_2" date="2012-10-28">
    <name>HP</name>
  </manufacture>
  <!-- Loop2 types -->
  <types model="business1">
    <type>DELL123</type>
    <manufacture_id>manu_1</manufacture_id>
  </types>
</products>
```
The objective of this scenario is to restructure the products data to streamline the presentation of the products information to serve the manufacturing operations. The expected output data is as follows. The root element is changed to manufacturers, the sales information is consolidated into the sale element, and the manufacturer element is reduced to one single level.

```xml
<?xml version="1.0" encoding="ISO-8859-15"?>
<manufacturers category="1" name="laptop">
  <sales unit="Dollars">
    <sale sales_type="DELL123">12345.0</sale>
    <sale sales_type="HP123">12345.123</sale>
  </sales>
  <manufacturer id="manu_1" date="03-04-0036" name="DELL"/>
  <manufacturer id="manu_2" date="04-04-0034" name="HP"/>
  <types>
    <type>DELL123</type>
    <manufacturer_id>manu_1</manufacturer_id>
  </types>
  <types>
    <type>DELL123</type>
    <manufacturer_id>manu_2</manufacturer_id>
  </types>
  <types>
    <type>HP123</type>
    <manufacturer_id>manu_1</manufacturer_id>
  </types>
  <types>
    <type>HP123</type>
    <manufacturer_id>manu_2</manufacturer_id>
  </types>
</manufacturers>
```

### Adding and linking the components

**Procedure**

1. Create a new Job and add a tFileInputXML component, a tXMLMap component, a tLogRow component, and a tFileOutputXML component by typing their names in the design workspace or dropping them from the Palette.

2. Link the tFileInputXML component to the tXMLMap component using a Row > Main connection.

3. Link the tXMLMap component to the tLogRow component using a Row > “New Output” (Main) connection. In the pop-up dialog box, enter the name of the output connection, outDoc in this example.

4. Link the tLogRow component to the tFileOutputXML component using a Row > Main connection.
Configuring the input flow

Procedure

1. Double-click the tFileInputXML component to open its Basic settings view.

2. Click the [...] button next to Edit schema and in the [Schema] dialog box define the schema by adding one column doc of Document type.

3. Click OK to validate the changes and close the dialog box. One row is added automatically to the Mapping table.

4. In the File name/Stream field, browse to or type in between double quotation marks the path to the XML source file that provides the products data. In this scenario, it is E:/ProductsIn.xml.

5. In the Loop XPath query field, type in an XPath expression between double quotation marks to specify the node on which the loop is based. In this scenario, it is /, which means to perform look query from the root.

6. In the XPath query column of the Mapping table, type in the fields to be queried between double quotation marks. In this scenario, it is ., which means all fields under the current node (root) will be extracted.

7. In the Get Nodes column of the Mapping table, select the check box.
Configuring tXMLMap with multiple loops

Procedure

1. Double-click the tXMLMap component to open its Map Editor.

![Image of tXMLMap Map Editor]

Note that the input area is already filled with the default basic XML structure and the top table is the main input table.

2. In the row1 input table, right-click the doc node and from the contextual menu select Import From File. In the pop-up dialog box, browse to the XML source file to import therefrom the XML structure used by the data to be received by tXMLMap. In this scenario, the XML source file is ProductsIn.xml, which contains the input data to tFileInputXML.

![Image of XML tree with doc node]

3. In the imported XML tree, right-click the manufacturer node and from the contextual menu select As loop element to set it as the loop element. Then do the same to set the types node and the sale node as loop elements respectively.
4. On the lower part of the map editor, click the **Schema editor** tab to display the corresponding view. Then on the right side of this view, add one column `outDoc` of **Document** type to the schema table. The corresponding XML root is added automatically to the output table on the top right side which represents the output flow.

5. In the `outDoc` output table, import the XML data structure to be used from the XML file that contains the expected output data and provides the expected XML structure. Right-click the `sale` node in the output table and select **As loop element** from the contextual menu. Then do the same to set the `manufacturer` node and the `types` node as loop elements respectively.
6. In the row1 input table, click the @category node and drop it to the Expression field of the @category node in the outDoc output table.

Do the same to map other nodes from the input table to the output table:

- the @name node to the @name node,
- the @unit node under the summary node to the @unit node,
- the @id node to the @id node and to the manufacturer_id node respectively,
- the @date node to the @date node,
• the name node to the @name node,
• the type node to the type node,
• the @type node to the @sales_type node, and
• the income node to the sale (loop) node.

7. On the top of the outDoc output table, click the wrench icon and set the value of the All in one property to true to generate a single XML flow. For further information about the All in one feature, see Talend Studio User Guide.

8. Click the [...] button next to the manufacturer loop element and in the pop-up [Configure source loops] dialog box click the [+] button to add one source loop manufacturer. Do the same to add one source loop sale for the sale loop element.

9. Click the [...] button next to the types loop element and in the pop-up [Configure source loops] dialog box add two source loops types and manufacturer. Make sure the sequence number of the types source loop is 0 so that the relative part of the output flow will be sorted based on the values of the type element.

Note:
When a loop element receives mappings from more than one loop element of the input flow, it allows you to set the sequence of the input loops. For example, the types loop element of the output flow in this scenario is mapped with the @id node which belongs to
the manufacturer loop element and the type node which belongs to the types loop element of the input flow. The output flow will be sorted according to the primary types loop.

10. Click OK to validate the mappings and close the **Map Editor**.

### Configuring the output flow

#### Procedure

1. Double-click the tLogRow component to open its **Basic settings** view.
2. Click the **Sync columns** button to retrieve the schema from its preceding component and accept the propagation prompted by the pop-up dialog box.
3. Double-click the tFileOutputXML component to open its **Basic settings** view.

![tFileOutputXML_1](image)

4. In the **File Name** field, browse to or enter the path to the file in which the output data will be written. In this scenario, it is *E:/ProductsOut.xml*.
5. Select the **Incoming record is a document** check box.

### Saving and executing the Job

#### Procedure

1. Press **Ctrl+S** to save the Job.
2. Press **F6** to execute the Job.
As shown above, the input products data is restructured as expected and the output data is displayed on the console and written into the XML file `ProductsOut.xml`.

```xml
<?xml version="1.0" encoding="ISO-8859-15"?>
<manufacturers category="1" name="laptop">
  <sale sales_type="DELL123">12345.0</sale>
  <sale sales_type="HP123">12345.123</sale>
  <manufacturer id="manu_1" date="03-04-0036" name="DELL"/>
  <manufacturer id="manu_2" name="HP"/>
</manufacturers>
```
**tXMLRPCInput**

Invokes a Method through a Web service and for the described purpose.

This component calls the defined method from the invoked RPC service, and returns the class as defined, based on the given parameters.

**tXMLRPCInput Standard properties**

These properties are used to configure tXMLRPCInput running in the Standard Job framework.

The Standard tXMLRPCInput component belongs to the Internet family.

The component in this framework is generally available.

**Basic settings**

| Schema and Edit Schema | A schema is a row description, it defines the number of fields that will be processed and passed on to the next component. The schema is either Built-in or remote in the Repository. Click Edit schema to make changes to the schema. If the current schema is of the Repository type, three options are available:  
  * View schema: choose this option to view the schema only.  
  * Change to built-in property: choose this option to change the schema to Built-in for local changes.  
  * Update repository connection: choose this option to change the schema stored in the repository and decide whether to propagate the changes to all the Jobs upon completion. If you just want to propagate the changes to the current Job, you can select No upon completion and choose this schema metadata again in the [Repository Content] window.  
In the RPC context, the schema corresponds to the output parameters. If two parameters are meant to be returned, then the schema should contain two columns. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Server URL</td>
<td>URL of the RPC service to be accessed</td>
</tr>
<tr>
<td>Need authentication / Username and Password</td>
<td>Select this check box and fill in a username and password if required to access the service.</td>
</tr>
<tr>
<td>Method Name</td>
<td>Enter the exact name of the Method to be invoked. The Method name MUST match the corresponding method described in the RPC Service. The Method name is also case-sensitive.</td>
</tr>
<tr>
<td>Return class</td>
<td>Select the type of data to be returned by the method. Make sure it fully matches the one defined in the method.</td>
</tr>
</tbody>
</table>
Parameters

Enter the parameters expected by the method as input parameters.

Global Variables

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

**NB_LINE**: the number of rows processed. This is an After variable and it returns an integer.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see [Talend Studio User Guide](#).

Usage

| Usage rule | This component is generally used as a Start component. It requires to be linked to an output component. |

Scenario: Guessing the State name from an XMLRPC

This scenario describes a two-component Job aiming at using a RPC method and displaying the output on the console view.

- Drop the **tXMLRPCInput** and a **tLogRow** components from the **Palette** to the design workspac.
- Set the **tXMLRPCInput** basic settings.
• Define the **Schema type** as **Built-in** for this use case.

• Set a single-column schema as the expected output for the called method is only one parameter: `StateName`.

• Then set the **Server url**. For this demo, use: `http://phpxmlrpc.sourceforge.net/server.php`

• No authentication details are required in this use case.

• The **Method** to be called is: `examples.getStateName`

• The **Return class** is not compulsory for this method but might be strictly required for another. Leave the default setting for this use case.

• Then set the input **Parameters** required by the method called. The **Name** field is not used in the code but the value should follow the syntax expected by the method. In this example, the Name used is **State Nr** and the value randomly chosen is **42**.

• The class has not much impact using this demo method but could have with another method, so leave the default setting.

• On the **tLogRow** component **Component** view, check the box: **Print schema column name in front of each value**.

• Then save the Job and press **F6** to execute it.

```
StateName: South Dakota
Job xmlrpc ended at 16:25 21/09/2007. [exit code=0]
```

*South Dakota* is the state name found using the `getStateName` RPC method and corresponds the 42nd State of the United States as defined as input parameter.
tXSDValidator

Helps at controlling data and structure quality of the file or flow to be processed.

tXSDValidator validates an input XML file or an input XML flow against an XSD file and sends the validation log to the defined output.

**tXSDValidator Standard properties**

These properties are used to configure tXSDValidator running in the Standard Job framework. The Standard tXSDValidator component belongs to the XML family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Mode**           | Select the validation mode from the drop-down list.  
  - **File Mode**: to validate an input file.  
  - **Flow Mode**: to validate an input flow. |
| **Schema and Edit schema** | A schema is a row description. It defines the number of fields to be processed and passed on to the next component.  
  Note that when **File Mode** is selected from the **Mode** list, the schema of this component is read-only and it contains standard information regarding the file validation. |
| **XSD file**       | Specify the path to the XSD reference file. The HTTP URL is also supported, for example, `http://localhost:8080/book.xsd`.  
  This field is available only when **File Mode** is selected from the **Mode** drop-down list. |
| **XML file**       | Specify the path to the XML file to be validated.  
  This field is available only when **File Mode** is selected from the **Mode** drop-down list. |
| **If XML is valid, display** | Type in the message to be displayed on the console if the XML file is valid.  
  This field is available only when **File Mode** is selected from the **Mode** drop-down list. |
| **If XML is invalid, display** | Type in the message to be displayed on the console if the XML file is invalid.  
  This field is available only when **File Mode** is selected from the **Mode** drop-down list. |
| **Print to console** | Select this check box to display the validation message on the console. |
This check box is available only when **File Mode** is selected from the **Mode** drop-down list.

### Allocate

Click the [+] button to add as many rows as needed, and in each row set the value of the following columns:

- **Input Column**: click the cell and select a column to be validated.
- **XSD File**: enter the path to the corresponding XSD reference file.

This table is available only when **Flow Mode** is selected from the **Mode** drop-down list.

### Advanced settings

#### Enable Features

Click the [+] button to add as many rows as needed, and in each row enter the feature to be enabled on the underlying parser between double quotation marks, for example, "http://apache.org/xml/features/honour-all-schemaLocations".

For more information about the features, see [https://xerces.apache.org/xerces2-j/features.html](https://xerces.apache.org/xerces2-j/features.html).

#### Encoding

Enter the encoding type between double quotation marks.

#### tStatCatcher Statistics

Select this check box to gather the Job processing metadata at the Job level as well as at each component level.

### Global Variables

**Global Variables**

**ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the **Die on error** check box is cleared, if the component has this check box.

**DIFFERENCE**: the result of the validation. This is a Flow variable and it returns a string.

**VALID**: the validation result. This is a Flow variable and it returns a boolean.

**XSD_ERROR_MESSAGE**: the xsd error message generated by the component. This is a Flow variable and it returns a string.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.

For further information about variables, see *Talend Studio User Guide*.
Usage

| Usage rule | When **File Mode** is selected, this component can be used as a standalone component but it is usually linked to an output component to gather the log data. |

**Scenario: Validating data flows against an XSD file**

This scenario describes a Job that validates an XML column in the input file *ShipOrder.csv* against the XSD reference file *ShipOrder.xsd* and then outputs valid rows into the delimited file *ShipOrder_Valid.csv* and invalid rows and error messages into the delimited file *ShipOrder_Invalid.csv*. For a similar use case that validates an XML file, see Validating XML files on page 663.

The content of the input file *ShipOrder.csv* that includes the XML column *ShipOrder* to be validated is as follows:

```
ID;ShipOrder
000001;<shiporder orderid="000001"><orderperson>George Bush</orderperson><shipto>John Adams</shipto><item><title>Empire Burlesque</title><note>Special Edition</note><quantity>1</quantity><price>10.90</price></item></shiporder>
000002;<shiporder orderid="000002"><orderperson>Judy Liu</orderperson><shipto>Jack Liu</shipto><item><title>Hide Your Heart</title><quantity>1</quantity><price>9.90</price></item></shiporder>
000003;<shiporder><orderperson>Peter Qian</orderperson><shipto>Thomas Wang</shipto><item><title>The Power of Habit</title><quantity>1</quantity><price>8.99</price></item></shiporder>
```

The content of the XSD reference file *ShipOrder.xsd* is as follows:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="shiporder">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="orderperson" type="xs:string"/>
        <xs:element name="shipto">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="name" type="xs:string"/>
              <xs:element name="address" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
Adding and linking components

Procedure
1. Create a new Job and add a tFileInputDelimited component, a tXSDValidator component, and two tFileOutputDelimited components by typing their names in the design workspace or dropping them from the Palette.

2. Double-click the tXSDValidator component to open its Basic settings view and select Flow Mode from the Mode drop-down list.

3. Link the tFileInputDelimited component to the tXSDValidator component using a Row > Main connection.

4. Link the tXSDValidator component to the first tFileOutputDelimited component using a Row > Main connection to output valid rows.

5. Link the tXSDValidator component to the second tFileOutputDelimited component using a Row > Rejects connection to output invalid rows.

Configuring the components

Procedure
1. Double-click the tFileInputDelimited component to open its Basic settings view on the Component tab.

2. In the File name/Stream field, specify the path to the input file. In this example, it is E:/ShipOrder.csv.
In the **Header** field, enter 1 to skip the first header row of the input file.

Click the [...] button next to **Edit schema** and define the schema by adding two columns **ID** and **ShipOrder** of **String** type.

3. Double-click the **tXSDValidator** component to open its **Basic settings** view on the **Component** tab.

4. Click the **Sync columns** button to retrieve the schema from the preceding **tFileInputDelimited** component, and in the pop-up dialog box, click **Yes** to propagate the schema to the two **tFileOutputDelimited** components.

Add a row in the **Allocate** table by clicking the [+] button. Then click the **Input Column** cell and select the XML column **ShipOrder** to be validated from the drop-down list. And in the **XSD File** cell, enter the path to the XSD reference file, *E:/ShipOrder.xsd* in this example.

5. Double-click the first **tFileOutputDelimited** component to open its **Basic settings** view on the **Component** tab.
6. In the File Name field, specify the path to the output file that will store valid rows. In this example, it is E:/ShipOrder_Valid.csv. Select the Include Header check box to include column headers in the output file.

7. Double-click the second tFileOutputDelimited component to open its Basic settings view on the Component tab.

8. Click the [...] button next to Edit schema to view its schema.

You can see an extra column errorMessage that holds the error information for invalid rows is added automatically into the schema in addition to the two propagated columns.

9. In the File Name field, specify the path to the output file that will store invalid rows and error messages. In this example, it is E:/ShipOrder_Invalid.csv. Select the Include Header check box to include column headers in the output file.

Saving and executing the Job

Procedure

1. Press Ctrl+S to save the Job.
2. Press F6 to run the Job.
As shown above, the output file *ShipOrder_Valid.csv* contains two valid rows, and the output file *ShipOrder_Invalid.csv* contains one invalid row that doesn’t define the *orderid* attribute and the error message.
tXSLT

Helps to transform data structure to another structure.
This component refers to an XSL stylesheet, to transform an XML source file into a defined output file.

**tXSLT Standard properties**

These properties are used to configure tXSLT running in the Standard Job framework.
The Standard tXSLT component belongs to the XML family.
The component in this framework is generally available.

**Basic settings**

<table>
<thead>
<tr>
<th>XML file</th>
<th>File path to the XML file to be validated.</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSL file</td>
<td>File path to the reference XSL transformation file.</td>
</tr>
<tr>
<td>Output file</td>
<td>File path to the output file. If the file does not exist, it will be created. The output file can be any structured or unstructured file such as html, xml, txt or even pdf or edifact depending on your xsl.</td>
</tr>
</tbody>
</table>

**Parameters**

Click the plus button to add new lines in the Parameters list and define the transformation parameters of the XSLT file. Click in each line and enter the key in the name list and its associated value in the value list.

**Advanced settings**

| tStatCatcher Statistics | Select this check box to gather the processing metadata at the Job level as well as at each component level. |

**Global Variables**

- **ERROR_MESSAGE**: the error message generated by the component when an error occurs. This is an After variable and it returns a string. This variable functions only if the Die on error check box is cleared, if the component has this check box.

- **OUTPUT_FILEPATH**: the output file path. This is a Flow variable and it returns a string.

- **OUTPUT_FILENAME**: the output file name. This is a Flow variable and it returns a string.

A Flow variable functions during the execution of a component while an After variable functions after the execution of the component.

To fill up a field or expression with a variable, press **Ctrl + Space** to access the variable list and choose the variable to use from it.
For further information about variables, see Talend Studio User Guide.

Usage

<table>
<thead>
<tr>
<th>Usage rule</th>
<th>This component can be used as standalone component.</th>
</tr>
</thead>
</table>

Limitation

Due to license incompatibility, one or more JARs required to use this component are not provided. You can install the missing JARs for this particular component by clicking the Install button on the Component tab view. You can also find out and add all missing JARs easily on the Modules tab in the Integration perspective of your studio. You can find more details about how to install external modules in Talend Help Center (https://help.talend.com).

Scenario: Transforming XML to html using an XSL stylesheet

This scenario describes a two-component Job that converts xml data into an html document using an xsl stylesheet. It as well defines a transformation parameter of the xsl stylesheet to change the background color of the header of the created html document.

Procedure

Procedure

1. Drop the tXSLT and tMsBox components from the Palette to the design workspace.

2. Double-click tXSLT to open its Basic settings view where you can define the component properties.
3. In the **XML file** field, set the path or browse to the xml file to be transformed. In this example, the xml file holds a list of MP3 song titles and related information including artist names, company etc.

4. In the **XSL file** field in the **Basic settings** view, set the path or browse to the relevant xsl file.

5. In the **Output file** field, set the path or browse to the output html file.

In this example, we want to convert the xml data into an html file holding a table heading followed by a table listing artists' names next to song titles.
6. In the Parameters area of the Basic settings view, click the plus button to add a line where you can define the name and value of the transformation parameter of the xsl file. In this example, the name of the transformation parameter we want to use is bgcolor and the value is green.

7. Double-click the tMsgBox to display its Basic settings view and define its display properties as needed.

8. Save the Job and press F6 to execute it. The message box displays confirming that the output html file is created and stored in the defined path.
9. Click **OK** to close the message box.

**Results**

You can now open the output html file to check the transformation of the xml data and that of the background color of the table heading.

**My CD Collection**

<table>
<thead>
<tr>
<th>Title</th>
<th>Artist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empire Burlesque</td>
<td>Bob Dylan</td>
</tr>
<tr>
<td>Hide your heart</td>
<td>Bonnie Tyler</td>
</tr>
<tr>
<td>Greatest Hits</td>
<td>Dolly Parton</td>
</tr>
<tr>
<td>Still got the blues</td>
<td>Gary Moore</td>
</tr>
</tbody>
</table>